This Bash script serves as a practical demonstration of Infrastructure as Code (IaC) principles in OpenStack. By codifying the steps required to provision a network, configure security, launch a virtual machine (VM), and install Nginx, the script automates tasks that would otherwise require manual dashboard interaction. As such, it highlights the fundamental value of automation in cloud operations: speed, consistency, auditability, and safety.

Speed and repeatability. Automation reduces the time required to create infrastructure. The script provisions a network, security group, key pair, and VM in a single run, with a companion teardown function for rapid decommissioning. Empirical research shows that IaC reduces provisioning time and enables faster delivery cycles (Rahman et al., 2019; Wiedemann, Lesch and Bogner, 2022).

Consistency and reduced error. By applying the exact instructions every time, scripting mitigates configuration drift. For example, the script restricts ingress traffic to SSH and HTTP only, ensuring a consistent security baseline. Repeated runs yield predictable environments, reducing human error, a recurring issue in manual configuration (Rahman et al., 2019).

Auditability and governance. Infrastructure expressed as code can be stored in version control, peer-reviewed, and subject to automated testing. This script incorporates parameterised variables and idempotent checks (e.g., “create if not exists”), supporting transparent governance in line with DevOps principles (Palomba et al., 2017; Wiedemann, Lesch and Bogner, 2022).

Safety and lifecycle control. By including teardown and validation routines, the script helps prevent resource sprawl and supports operational hygiene. This responds to risks highlighted by Chiari et al. (2022), who identify that IaC, like application code, may introduce defects with real-world consequences if not managed.

**Critical perspective**

While the benefits are clear, there are limitations and trade-offs to using Bash scripting as an IaC mechanism.

Imperative vs declarative. Bash scripts outline the step-by-step process for creating resources. This provides control but can be verbose and less scalable than declarative IaC tools (e.g., Terraform, Heat), which instead define what the final state should look like (Rahman et al., 2019). As environments grow, declarative approaches generally reduce complexity.

Error handling and idempotence. Although this script checks for existing resources, Bash lacks inherent state management. Declarative tools maintain infrastructure state and can automatically converge towards the desired configuration, making them more reliable in enterprise-scale contexts (Wiedemann, Lesch, and Bogner, 2022).

Maintainability. Bash is widely available and straightforward, but it can be less maintainable in larger teams or complex systems. For example, multi-cloud or multi-region architectures would require additional scripts, increasing operational overhead. Studies suggest that tool choice significantly affects productivity and defect rates in IaC projects (Chiari et al., 2022).

Thus, while Bash scripting is valuable for educational settings, prototypes, and lightweight automation, organisations at scale often migrate towards specialised IaC platforms to balance control with maintainability. Critically, this reflects the broader lifecycle of IaC adoption, starting with scripts and then moving towards declarative frameworks integrated into CI/CD pipelines.

**Conclusion**

In summary, the Bash script demonstrates the tangible benefits of IaC fast, consistent, auditable cloud provisioning, while also exposing the challenges of relying on imperative scripting at scale. It therefore serves as a stepping stone for understanding IaC concepts and a bridge to more advanced frameworks, highlighting both the strengths and limitations of scripting automation in cloud operations.

**References**

Chiari, M., Di Biase, M., Leppänen, V. and Palomba, F. (2022) ‘Infrastructure as Code: The state of research and practice’, *arXiv preprint* arXiv:2206.10344. Available at: https://arxiv.org/abs/2206.10344 (Accessed: 17 August 2025).

Palomba, F., Tamburri, D.A., Serebrenik, A. and Zaidman, A. (2017) ‘Infrastructure as Code: A survey and future directions’, in *ICSE NIER*. IEEE, pp. 341–344. Available at: https://fpalomba.github.io/pdf/Conferencs/C42.pdf (Accessed: 17 August 2025).

Rahman, M.A., Williams, L., Barik, T. and Parnin, C. (2019) ‘A systematic mapping study of Infrastructure as Code research’, *Information and Software Technology*, 108, pp. 65–77. doi: 10.1016/j.infsof.2018.12.004.

Wiedemann, A., Lesch, V. and Bogner, J. (2022) ‘Infrastructure as Code: A survey on the state of practice’, *Empirical Software Engineering*, 27(3), pp. 1–37. doi: 10.1007/s10664-021-10053-5.