The Right way to structure Terraform Project!

Atul Anand · Follow

5 min read · Sep 30, 2023

176 2

The Right way to structure Terraform Project

Crafting a thriving Terraform project goes beyond technical skills; it calls for a strategic blend of expertise and precision. With this article,

let me guide you through the systematic approach to structure your Terraform project, outlining fundamental coding principles that should be adhered to and sharing invaluable best practices that ensure the project's efficiency and scalability for the future.

If you haven't already, I would recommend checking out the ***last blog*** of this series, where we looked at an extensive list of all the prominent Infrastructure-as-Code tools and compared them thoroughly.

10 Essential Aspects of a Terraform Project

When working on a Terraform project, it's essential to take into account a range of factors that contribute to its effectiveness. The top 10 factors include:

1. **Project Structure:** Maintain a standardized format for files and

increase code reusability.

3. **Naming Conventions:** Implement naming conventions for configuration objects, such as variables, modules, and files, to ensure uniformity in resource names.
4. **Code Formatting and Style:** Employ consistent coding practices to keep your code clean, easy to read, understand, and maintain.
5. **Variables Declaration and Assignment:** Avoid hard-coding values to ensure code reusability and organization, especially for future modifications.

6. **Module Outputs:** Effectively integrate modules and the resources contained within them.
7. **Data Sources:** Leverage data sources to access resources defined outside of the Terraform project.
8. **Remote State Management:** Facilitate collaboration within teams and safeguard the state file from accidental deletion.
9. **Project Blast Radius:** Maintain an optimal project size, grouping related resources together while striking a balance between coupling and cohesion.
10. **Version Control:** Implement code version control to efficiently manage different versions of the code, facilitating collaboration and code sharing within teams.

Structuring Terraform Project

Determining the division of the Terraform code is inherently subjective. The structure of a Terraform project can vary significantly based on the project's size and specific requirements. In the case of a straightforward project, it's feasible to consolidate everything into a single file or just a couple of files within a single folder, simplifying the task.

However, in the context of a well-architected project, scalability is a key consideration. The structure should be designed to accommodate future growth without necessitating a complete overhaul. In this regard, there is a set of fundamental principles that should be considered when making decisions about the project's organization.

For ***simple, small-scale projects***, a good way to structure the Terraform project could be:

```
projectname/
    |
    |-- provider.tf
    |-- version.tf
    |-- backend.tf
    |-- main.tf
    |-- variables.tf
    |-- terraform.tfvars
    |-- outputs.tf
    |
    |-- modules/
    |   |
    |   |-- network/
    |   |   |
    |   |   |-- provider.tf
    |   |   |-- vpc.tf
    |   |   |-- loadbalancer.tf
    |   |   |-- variables.tf
    |   |   |-- variables-local.tf
    |   |   |-- outputs.tf
    |   |   |-- README.md
    |   |   |-- examples/
    |   |   |-- docs/
    |   |
    |   |-- vm/
    |        |
    |        |-- ...
    |
    |-- scripts/
    |-- files/
    |-- templates/
    |-- ...
```

For a ***complex project*** (of higher scale) spanning multiple environments and regions, one of the decent ways of organizing the project could be:

```
    projectname/
    |
    |-- environment/
    |   |
    |   |-- prod/
```

```
|   |   |
|   |   |-- region1/
|   |   |   |
|   |   |   |-- provider.tf
|   |   |   |-- version.tf
|   |   |   |-- backend.tf
|   |   |   |-- main.tf
|   |   |   |-- variables.tf
|   |   |   |-- terraform.tfvars
|   |   |   |-- outputs.tf
|   |   |
|   |   |-- region2/
|   |   |   |
|   |   |   |-- ...
|   |
|   |-- dev/
|   |   |
|   |   |-- ...
|   |
|   |-- stage/
|       |
|       |-- ...
|
|-- modules/
|   |
|   |-- network/
|   |   |
|   |   |-- provider.tf
|   |   |-- network.tf
|   |   |-- loadbalancer.tf
|   |   |-- variables.tf
|   |   |-- variables-local.tf
|   |   |-- outputs.tf
|   |   |-- README.md
|   |   |-- examples/
|   |   |-- docs/
|   |
|   |-- vm/
|       |
|       |-- ...
|
|-- scripts/
|-- files/
|-- templates/
|-- ...
```

Notes:

1. In principle, the nesting of folders should not go deeper than 3–4 levels, if possible, in order to keep the complexity of the project less.
2. These are just some recommendations based on best practices and can be modified to inculcate any changes as per the requirements and convenience.

Understanding Terraform Project Components

Main Code Directory

The main code directory is the place (project root directory) from where we execute the Terraform commands and the code execution and deployment start. It mainly comprises these files:

```
|
|-- provider.tf
|-- version.tf
|-- backend.tf
|-- main.tf
|-- variables.tf
|-- terraform.tfvars
|-- outputs.tf
```

- `provider.tf` : Specifies the cloud or SaaS providers utilized.
- `version.tf` : Establishes Terraform version compatibility.
- `backend.tf` : Defines the backend configuration for state storage.
- `main.tf` : Serves as the central Terraform file from which modules are invoked.

- `variables.tf` : Contains variable declarations.
- `terraform.tfvars` : Stores variable definitions and assignments.
- `outputs.tf` : Includes output value definitions.

For multi-environment and multi-region deployments, which is usually the case in real-world projects, it is preferred to have an `environment` or `env` directory within the project with multiple environments like `dev`, `stage`, `prod`, etc. Furthermore, you can choose to create a `region` directory inside to separate the code and configurations for multiple regions.

Modules Directory

Modules represent a fundamental aspect of Terraform that empowers the creation of reusable infrastructure deployment code. Think of them as containers, bundling together multiple resources for cohesive usage.

Within the modules directory, we establish distinct subprojects of Terraform, each capable of being reused seamlessly from the root Terraform directory.

The key files and subdirectories within a module directory encompass:

```
|
|-- provider.tf
|-- resourcegroupname1.tf
|-- resourcegroupname2.tf
|-- variables.tf
|-- variables-local.tf
|-- outputs.tf
|-- README.md
```

```
|-- examples/
|-- docs/
```

- `provider.tf` : Specifies the provider employed for resources within the module.
- `<resourcegroupname>.tf` : Actual Terraform file(s) containing resource definitions for a particular group of resources, often organized into separate files for clarity.
- `variables.tf` : Contains declarations for module-specific variables.
- `variables-local.tf` : Utilized for local variables exclusive to the module.
- `outputs.tf` : Comprises definitions for output values.
- `README.md` : A comprehensive document detailing the purpose of the module, supported resource types, dependencies, and more.
- `examples/` : Reserved for storing illustrative code samples.
- `docs/` : A directory housing supplementary documents related to the module.

Other Directories

- `scripts/` : This directory serves as a repository for any custom scripts tailored to the project's specific requirements.
- `files/` : All non-executable static files referenced by Terraform are stored here, ensuring a central location for easy management and access.
- `templates/` : This directory is designated for files utilized by Terraform's "templateFile" function, streamlining the organization of template-related resources.

Conclusion

The initial planning phase of a Terraform project is of paramount importance in ensuring its ultimate success. Given that the project's structure significantly impacts scalability and reusability, it should be accorded top priority. Neglecting this crucial aspect in the early stages of the project could potentially hinder its long-term efficiency and effectiveness.

Please make sure you give this post **50 claps** 👏🏻 and my blog a **follow** ❤️ if you enjoyed it and want to see more.
Happy Learning! 🚀

Terraform | Project Structure | Best Practices | Iac

Written by Atul Anand

Follow

152 Followers

Software Engineer | Tech Evangelist | Full-stack Development | Cloud & DevOps
www.linkedin.com/in/ibatulanand

More from Atul Anand

Atul Anand in AWS in Plain English

Mirror GitHub repository to AWS CodeCommit using GitHub Actions

Simple 5-steps hands-on guide to mirror a GitHub Repository to an AWS CodeCommit...

5 min read · Oct 13, 2023

165

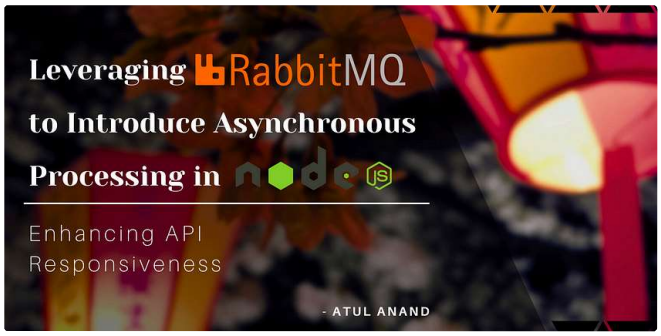
Atul Anand

Extensive Comparison of IaC tools in 2024!

The realm of Infrastructure as Code (IaC) tools presents an array of possibilities for...

4 min read · Jul 6, 2023

85



Atul Anand

Enhancing API Responsiveness: Leveraging RabbitMQ to Introduc...

In modern web applications, it is pretty common to encounter scenarios where...

7 min read · Jul 22, 2023

78

Atul Anand

Infrastructure as Code (IaC)—What is it & Do we really need it?

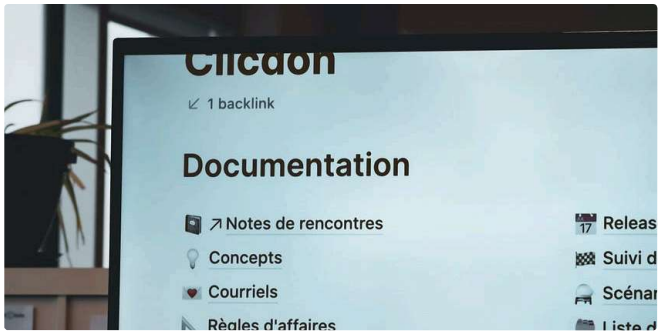
Infrastructure as Code??? Are you curious about its role in modern software...

4 min read · Apr 15, 2023

106

See all from Atul Anand

Recommended from Medium



Akhilesh Mishra

Terraform documentation made easy with terraform-docs

A complete guide to Terraform documentation with terraform-docs

6 min read · Nov 3, 2023

122

Javier Canizalez in Terrakube

Introducing Terrakube 2.19.0 🚀

OpenTofu Now Available on Terrakube

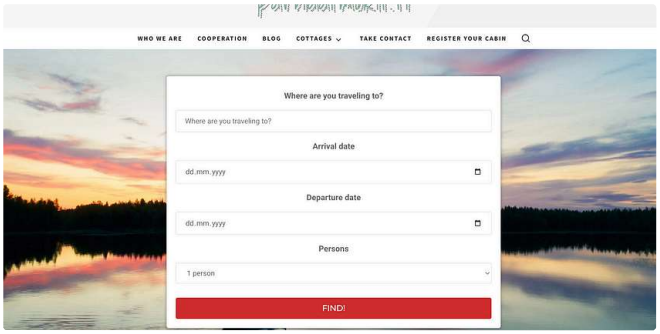
4 min read · Feb 2, 2024

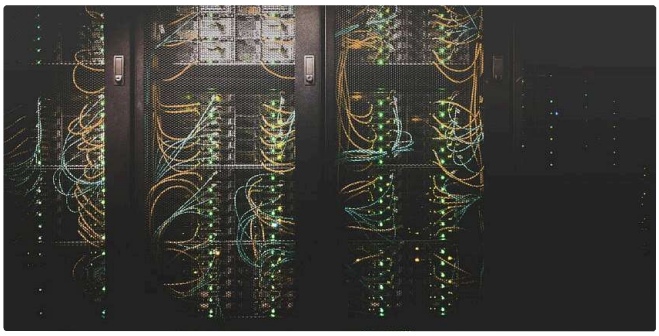
23 1

Lists

Natural Language Processing

1223 stories · 702 saves





Artturi Jalli

I Built an App in 6 Hours that Makes $1,500/Mo

Copy my strategy!

✨ · 3 min read · Jan 23, 2024

10.5K 138

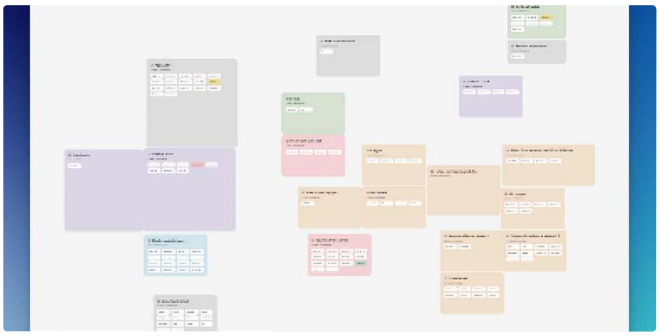
Jack Lindamood

(Almost) Every infrastructure decision I endorse or regret after ...

Assortment of technology start infrastructure rec

15 min read · Feb 7, 2024

877 19



AndyCheung0211

24 Productivity tools for 2024

introduce 24 tools for maximum productivity

✨ · 3 min read · Jan 3, 2024

2.2K 27

typo in Typo blog

Best CI/CD tools (2024)

This blog was originally published in the Typo blog.

10 min read · Jan 11, 2024

195 6

See more recommendations