

The Wizard's Guide to Well-Architected IaC Workloads

July 15, 2025



Who Am I?

Craig Reeder

- Senior Cloud Engineer @ Uturn Data Solutions
- Consulted for over a hundred companies, from startups to large enterprises
- Lead the internal Terraform/OpenTofu modules and practices
- Experience across AWS, Automation, DevOps, Kubernetes
- Hobby magician and mentalist



Requisite Knowledge / **Intended Audience**

- Assumed basic knowledge of Git or other source control tools
- Assumed basic knowledge of Terraform or other Infrastructure as Code (IaC) experience
 - If something needs additional explanation, feel free to ask!
- This is presented through the lens of AWS, but the principles apply to other cloud platforms as well



Which IaC Tool?

- OpenTofu is an open-source fork of Terraform due to a change in licensing in Terraform 1.6 (2023)
- OpenTofu is a CNCF & Linux Foundation Project
- I prefer OpenTofu over other IaC tools
 - Open-source, community led
 - Keeps things simpler than in a full language
 - Supports automation beyond AWS
- Everything in this presentation applies to both
- <https://cani.tf> – Excellent resource for OpenTofu/Terraform differences



Taming the Terraform Tornado



- The OpenTofu/Terraform community is still maturing and lacks the consistency found in established programming ecosystems
- It's a wild west of different patterns and practices
- Results: confusing and hard to maintain code bases, environmental drift
- "Terraform Workspaces" is a bad solution, and is on a path to deprecation
- This is an opinionated design that I believe addresses those problems
- Informed by:
 - Real world environments
 - Conversations with other engineers
 - Online discussions
 - Client feedback

Three Tier IaC – High Level

- Goals of this design:
 - Keep complexity low
 - Keep blast radius small
 - Reduce differences between environments
 - Allow for infrastructure versioning and promotion
 - Maintain DRY (Don't Repeat Yourself) as much as possible
 - OpenTofu/Terraform Only – No additional tooling necessary
- Three tiers of code: Module, Stack, Project
- Each tier should be in a separate repository
- Initial approach is “Infrastructure-Centric”



Tier One – Modules

- Represents distinct services, workload independent
- Primary goal is to simplify a service – what value does the module provide?
- Should reflect organizational opinions, such as default configurations
- Can be shared between teams
- Can use public modules like “Terraform AWS Modules” instead of maintaining your own
- Module repositories should be version tagged
- Examples of a module:
 - `aws_vpc`
 - `aws_s3_bucket`



Tier Two – Stacks



- Composition of modules and resources that represent a specific workload
- All resources created by a stack share a lifecycle
- Variables define differences between instances of the stack (i.e., scaling options, sizing, access policies)
- Stack repositories should be version tagged
- Examples of a stack:
 - `core_network`
 - `corp_website`
 - `github_runners`

Tier Three – Projects

- Thin wrapper – Instance of a stack
- This is where you run init and apply
- Contains provider and state configuration
- Passes environment-specific configurations as parameters to the referenced stack
- Does not contain variables itself – represents variables in environment
- For AWS, usually structured:
 - Top level folder: AWS Account
 - Second level folder: Region
 - Third+ level folder: Individual projects (organized with folders)



Overview – Putting it all together

Tier One: **Modules**

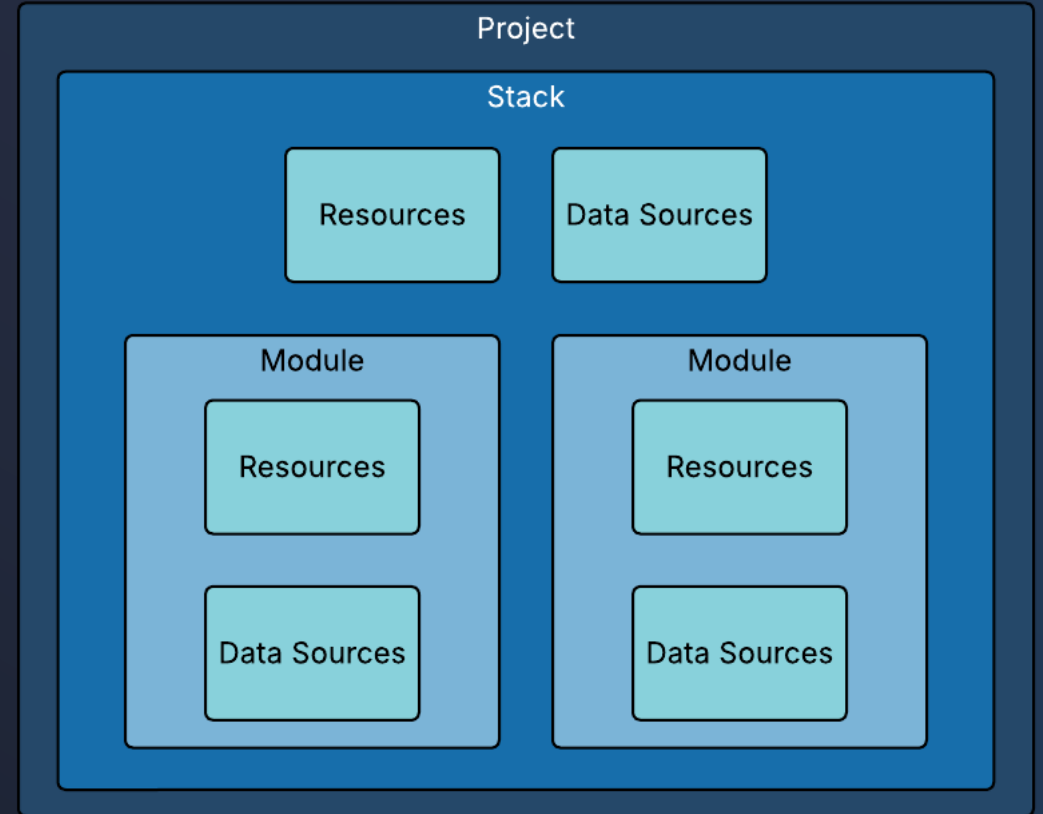
- Wraps distinct service or set of services for simplification

Tier Two: **Stacks**

- Composition of modules and resources to represent a workload

Tier Three: **Projects**

- Thin wrapper – instance of a stack for a given environment or account



Considerations – Mono-repo vs Poly-repo



- Each tier should exist in separate repositories
 - **Projects** reference tagged **Stacks** reference tagged **Modules**
- Any tier can be split into multiple repositories
- As with any technical decision, there are trade offs
 - All modules & stacks in the same repository share version tags
 - More repositories → higher management overhead
 - Less repositories → less semantic tags
- Generally, my clients do one repo for each tier initially

Considerations – Application-Centric IaC



- If you have application code that depends on specific infrastructure, it may be desirable to include the infrastructure dependencies with the application source code
- Can include project and/or stack in application repository
- Infrastructure and Application will share a version tag
- Use the same shared modules tier
- If you do this: make sure IaC is applied via CI/CD during application deployment
- This approach introduces additional considerations, particularly around environment promotion

Considerations – TACOS

- Tofu/Terraform Automation & Collaboration Software
- Advanced CI/CD for Infrastructure as Code
- Allows for teams to scale wider, more easily:
 - Can enforce policies, restrict access
 - Can manage drift detection
 - Enables automation around plan & apply workflows
- Requires design considerations
- Notable TACOS:
 - HCP Terraform (prev. Terraform Cloud/Enterprise)
 - Open Source: Atlantis, Digger, Terrateam
 - Spacelift, env0, Scalr



Additional Notes – Best Practices

- **Pick a pattern, and stick to it**
- Prefer resource-specific data sources over `terraform_remote_state`
- When developing modules or stacks locally, temporarily switch references to local paths for faster iteration
- Structure your local copy of repositories consistently
 - Automate repetitive tasks that are structure dependent
- Commit your `.terraform.lock.hcl` file in projects, but not in stacks and modules
- Run static security scanning at the project level
- Avoid using provisioners such as “local-exec” because they make modules machine specific



Additional Notes – Companion Repository

<https://github.com/creeder-uturn/wizardworkloads>

Includes:

- Example code for stacks and projects
- Use of “Terraform AWS Modules” open-source modules
- Boilerplate repository structure and dotfiles
- Links to additional reading and resources mentioned in slides
- Write-up on helpful CLI tools for local development
- PDF of this presentation



Who We Are

- Headquartered in Chicago IL
- Founded in 2013
- Cloud Consultancy
- Specializing in Cloud Migration, DevOps, Data & Analytics, and SaaS
- Near-Perfect AWS Customer Satisfaction Rating
- Servicing customers around the world
- #4 on Crain's Fast 50 fastest-growing companies in Chicago

- DevOps Services Competency
- Migration Competency
- SaaS Competency
- Data & Analytics Competency
- Immersion Day Partner
- Solution Provider
- AWS Lambda Delivery
- AWS Marketplace Seller
- Amazon Redshift Delivery Partner
- Well-Architected Partner Program
- Amazon EC2 for Windows Server Delivery
- Migration Acceleration Approved Partner
- Jumpstart Program Approved Partner
- Start-Up Migrate Program Approved Partner
- Public Sector Approved
- Data Foundation for GenAI
- AWS Strategic Collaboration Partner




Questions?

Craig Reeder, Senior Cloud Engineer

creeder@uturndata.com

uturndata.com

 [craig-reeder](#)



Companion Repository

The Uturn logo, featuring the word "Uturn" in a white sans-serif font, with a stylized cloud icon above the letter "u".