

ADD0

ALL DAY DEVOPS

NOVEMBER 12, 2020

Anton Babenko

How to build, scale,
and maintain 30 public
Terraform modules
with over 30 million
provisions



Anton Babenko

AWS Community Hero / HashiCorp Ambassador / Certified Terraform fanatic since 2015.

Organiser of HashiCorp UG, AWS UG, DevOps Norway, DevOpsDays Oslo.

I ❤️ open-source:

- ❏ [terraform-community-modules](#) + [terraform-aws-modules](#)
- ❏ [antonbabenko/pre-commit-terraform](#) — clean code, documentation, and more
- ❏ [serverless.tf](#) — Doing serverless on AWS with Terraform
- ❏ [antonbabenko/modules.tf-lambda](#) — generate Terraform code from visual diagrams
- ❏ [antonbabenko/terragrunt-reference-architecture](#) — Terragrunt reference architecture
- ❏ [www.terraform-best-practices.com](#)
- ❏ [bit.ly/terraform-youtube](#) — "Your weekly dose of #Terraform" & "Terraform tools review"
- ❏ @antonbabenko — Twitter, GitHub, LinkedIn



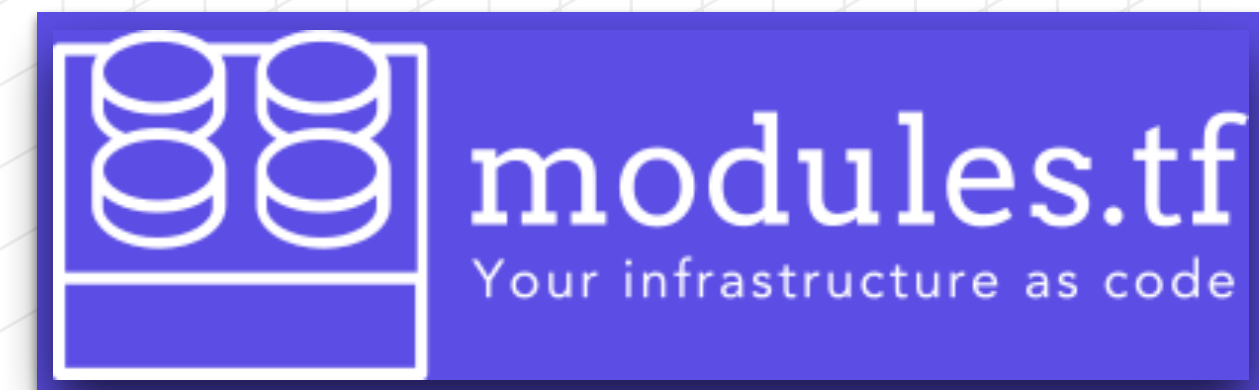
What do I do?

- ✓ All-things Terraform + AWS + DevOps
- ✓ Consulting
- ✓ Workshops
- ✓ Trainings
- ✓ Mentorship

My company: betajob.com

My blog: antonbabenko.com

My email: anton@antonbabenko.com



YOUR WEEKLY DOSE OF TERRAFORM

WITH ANTON BABENKO



@antonbabenko

```
module "terraform_review" {  
  source = "internet"  
  
  format = "weekly-live-stream"  
  content = ["news", "reviews", "live-coding", "Q&A"]  
}
```

Subscribe here — bit.ly/terraform-youtube

Your weekly dose of Terraform with news, reviews, Q&A, interviews, and live-coding.

@antonbabenko



SERVERLESS.TF

Build, deploy, and manage serverless applications
and infrastructure on AWS using Terraform

Collection of open-source Terraform AWS modules supported by the community with over 30 million provisions

(VPC, Autoscaling, RDS, Security Groups, ELB, ALB, Redshift, SNS, SQS, IAM, EKS, ECS, TGW, S3 bucket, CloudFront, Lambda, API Gateway, AppSync...)

github.com/terraform-aws-modules

registry.terraform.io/modules/terraform-aws-modules

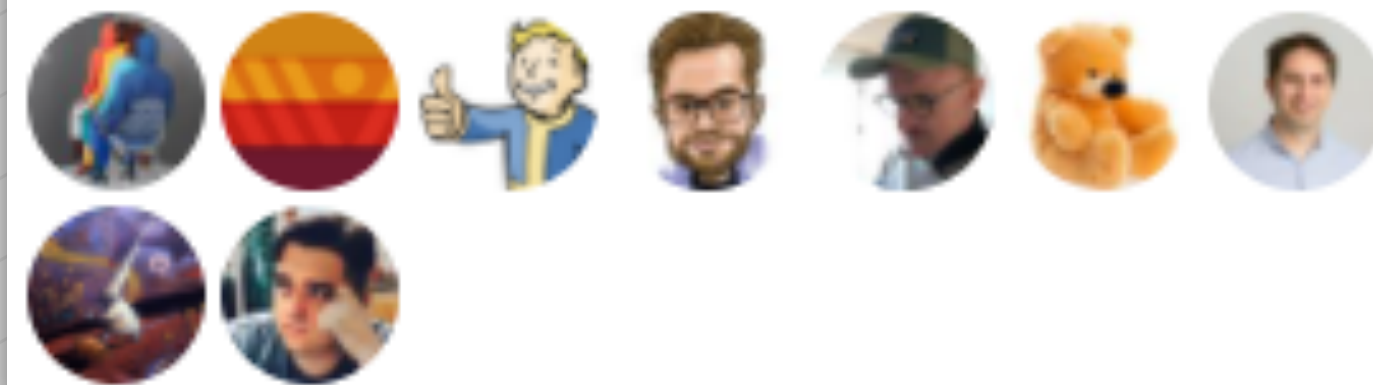
@antonbabenko



SERVERLESS.TF

Build, deploy, and manage serverless applications
and infrastructure on AWS using Terraform

Sponsors



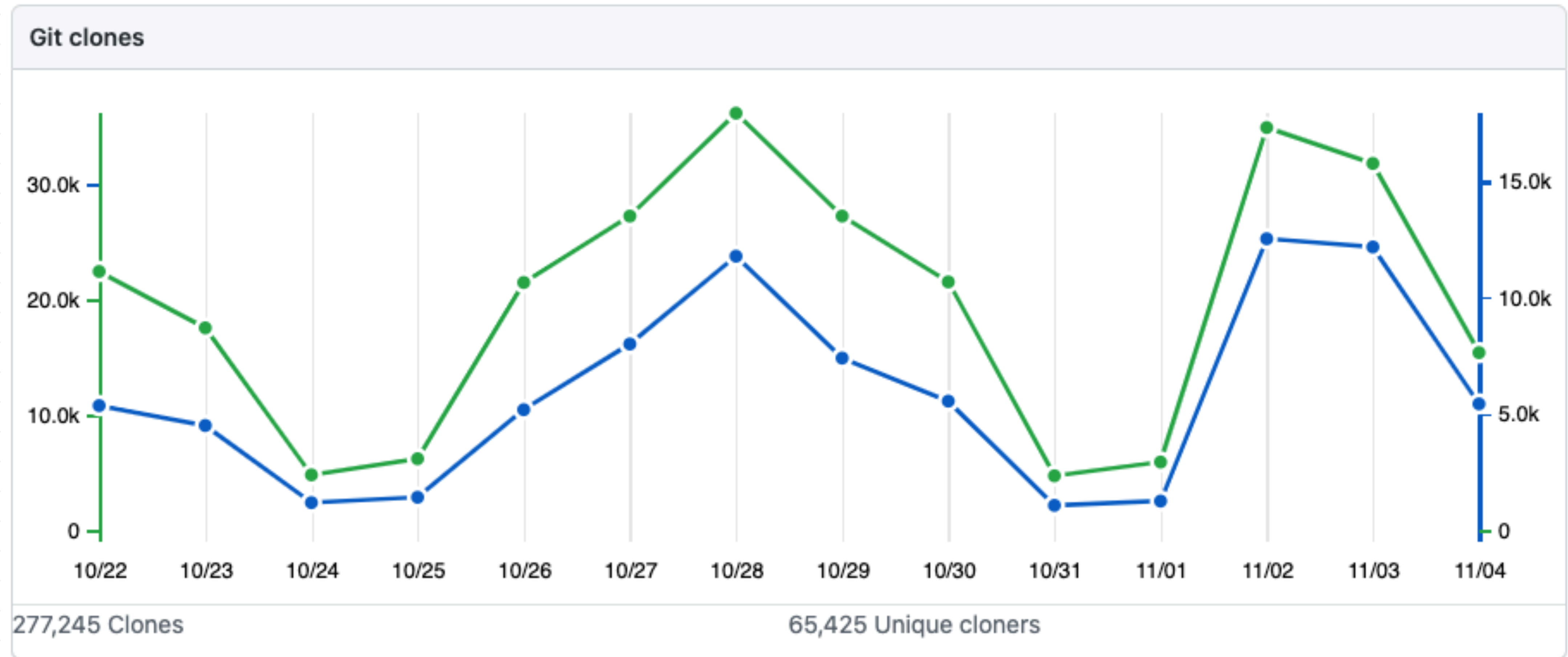
30+ million provisions, 1000+ pull-requests and issues resolved

Become a sponsor — github.com/sponsors/antonbabenko

Why do developers like terraform-aws-modules?

- ✓ Very large community
- ✓ Almost all features are shown in examples
- ✓ Documentation (by humans and pre-commit-terraform) — anything is better than nothing
- ✓ Modules compatibility and patterns used in each module between modules
- ✓ Week-zero support for the stable versions of Terraform

GitHub traffic — terraform-aws-vpc *



* this graph does not include usages from the Terraform Registry

Agenda

- 2015-2017 — How has it started?
- 2017 — Initial vision
- 2017-... — Scaling and maintaining
- 2020-... — Vision for the future
- Summary

How has it all started?

- ✓ github.com/terraform-community-modules
- ✓ tf_aws_vpc module was over-forked and contained similar changes
- ✓ Reusability aspect soon became rather obvious (to me, at least)
- ✓ Greenfield

Version 1 (2017)

- ❏ Terraform Registry launched (Terraform 0.10)
- ❏ Version 1 (my initial goals):
 - ❏ terraform-aws-modules = libraries
 - ❏ Simplify usage of AWS with Terraform
 - ❏ Create/migrate some existing modules (not strict BC guarantees)

Version 2 (2018)

- Version 2 goals:
- Hide the complexity inside of the modules (conditional creation of resources, no proper typing in variables, split/join, element/concat)
- Standardise names of variables and outputs — eg, `this_security_group_id`, all inputs and outputs. Published terraform-best-practices.com
- Changelog, Makefile, [pre-commit-terraform](#) hooks
- Consider adding testing (kitchen-terraform or terratest)

Scaling up

- ✦ A lot of feature requests for modules were rejected
- ✦ No compatibility and unified vision (lack of docs and examples), often single-user focus
- ✦ Importance of tooling (tflint, hub, semtag, code-reviews). Still WIP.
- ✦ Multiple versions support is very hard. When we can use TF 0.13 features and drop 0.11 support?

What is the module's scope for TAM?

- ✓ 100% flexibility (arguments and attributes)
- ✓ Cover the majority of common use-cases
- ✓ Require little maintenance (no provisions, no dependencies on other TF providers except logic, no jsonnet)
- ✓ Focus on resource modules to allow composition instead of bigger infrastructure modules (eg, CloudFront+S3+ACM instead of "module for static website")

terraform-aws-modules designs

- ✦ Define module's units, root module, submodules based on usage patterns (eg, one VPC with all resources VS one Security Group with rules)
- ✦ Use resources which are commonly used together (or BYOR)
- ✦ Parametrise everything and expose all outputs
- ✦ Do not hardcode anything user-specific
- ✦ Consider conditional creation of everything
- ✦ Use core features of Terraform (no workspaces, no overrides, no provisioners, no code generators)

Traits of good Terraform modules

- ✓ Documentation and examples
- ✓ Feature rich
- ✓ Sane defaults
- ✓ Clean code
- ✓ Tests

Read more: <http://bit.ly/common-traits-in-terraform-modules>

Maintaining & learning lessons

- ✦ Try to keep it simple, standardised, and invest in tools
- ✦ Reject even more "good" solutions with high complexity or used by a subset of the users
- ✦ Always imagine how user can use it (eg, examples, docs)
- ✦ Develop only what I or "real customers" may use and it is easy to maintain (TGW)
- ✦ Involve other contributors/maintainers/reviewers more (how?)

What about testing?

The reality in infrastructure testing



Anton Babenko 

AWS Community Hero / Terraform fanatic / HashiCorp Ambassador

3w • 



The best way of [#testing](#) in [#Terraform](#) now is to push to the master branch and wait half an hour for people who didn't pin version to open an issue.

Half an hour for the popular terraform-aws-module is enough. I know, I tried it.

PS: Here is a better way to do testing - <https://lnkd.in/dgTp8fE>



114 • 3 Comments

Why not Terratest for terraform-aws-modules?

- ✦ Expecting contributions from anyone (language, experience, background)
- ✦ Testing is for developers
- ✦ IaC Testing Pyramid
- ✦ Focus on static analysis (tflint, terraform validate)
- ✦ "terraform plan" + "terraform apply" on examples — is enough
- ✦ Infrastructure modules and compositions may benefit from Terratest

Infrastructure testing done right

- ✦ Use the same tools to assert the code (easy for users) — <https://github.com/apparentlymart/terraform-provider-testing/>
- ✦ "If the infrastructure is deployed it should have VPC in eu-west-1 with CIDR block 10.0.0.0/20" — <https://openvalidation.io/>
- ✦ Test-Driven Development for Infrastructure by Rosemary Wang — <https://github.com/joatmon08/tdd-infrastructure>

Future plans

Future plans




- ✦ Automation with GH actions
- ✦ Simplify codebase & refactor (eg, <https://github.com/minamijoyo/tfmigrate>)
- ✦ Document and verify the upgrade path for modules
- ✦ Experiment with everything-as-code — <https://github.com/terraform-aws-modules/meta>
- ✦ Continue with dog fooding
- ✦ We were missing serverless on AWS => <https://serverless.tf>

Summary: What can you do?

For your project/company

- ✦ Apply Amazon Leadership Principles (customer obsession, ownership, etc)
- ✦ Document intentions more (decisions = IaC)
- ✦ Write for users, not infrastructure experts
- ✦ Try to use existing solutions without copying/forking first
- ✦ Treat modules as library (leave complexity inside and expose simple interface)
- ✦ Add opinionated features if necessary

For OSS/community

-  Triage issues
-  Help others
-  Update tiny pieces of code at once

Thanks!

Questions?

Subscribe to "Your Weekly Dose of Terraform" — <http://bit.ly/terraform-youtube>

github.com/antonbabenko

twitter.com/antonbabenko

@antonbabenko