[Skip to content](https://dev.to/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij" \l "main-content)

[](https://dev.to/)

Top of Form

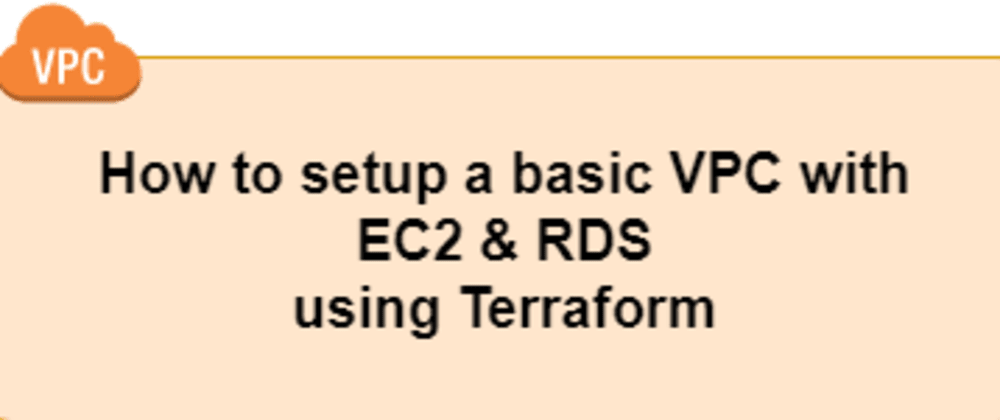


Bottom of Form

[Log in](https://dev.to/enter)[Create account](https://dev.to/enter?state=new-user)

19Add reaction

6Jump to Comments27Save

[](https://res.cloudinary.com/practicaldev/image/fetch/s--GMYNyZZp--/c_imagga_scale,f_auto,fl_progressive,h_420,q_auto,w_1000/https:/dev-to-uploads.s3.amazonaws.com/i/4bn5fhpwnfbmo9phcuus.png)

[Rolf Streefkerk](https://dev.to/rolfstreefkerk)

[Rolf Streefkerk](https://dev.to/rolfstreefkerk)

Posted on Mar 8, 2020

154

**How to setup a basic VPC with EC2 and RDS using Terraform**

[#aws](https://dev.to/t/aws)[#terraform](https://dev.to/t/terraform)[#devops](https://dev.to/t/devops)[#database](https://dev.to/t/database)

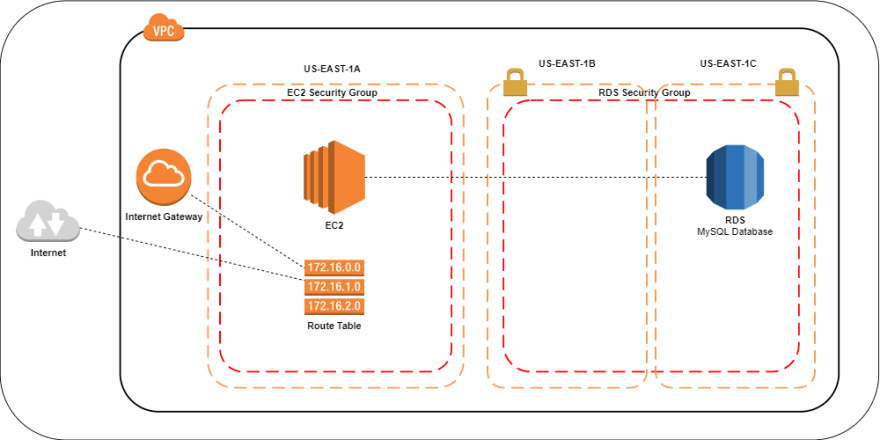
This guide will help you set up a basic AWS VPC with a virtual machine (EC2) and database (RDS) using Terraform (Infrastructure as Code).

I'll be breaking this topic down as follows:

* [The outline](https://dev.to/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij#the-outline)
* [VPC](https://dev.to/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij#vpc)
* [EC2](https://dev.to/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij#ec2)
  + [EC2 Security group](https://dev.to/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij#ec2-security-group)
* [RDS](https://dev.to/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij#rds)
  + [RDS Security group](https://dev.to/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij#rds-security-group)
* [Conclusion](https://dev.to/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij#conclusion)

**The outline**

We're going to create the following on AWS:

[](https://res.cloudinary.com/practicaldev/image/fetch/s--jpYNxJGI--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_880/https:/dev-to-uploads.s3.amazonaws.com/i/gc4m221kunlarpjlzoen.png)

A VPC with 1 Route table that connects the Internet Gateway to the public subnet that hosts the EC2 instance.

Two private subnets configured as 1 subnet group that hosts 1 RDS instance.

Access control is arranged using security groups, one for the EC2 public subnet and 1 for the RDS private subnets.

The reason we have 2 subnets for RDS is because that is a deployment requirement, you cannot launch an RDS instance without configuring it with 2 subnets.

Ideally, you would want to do load balancing for both EC2 and RDS instances. On the EC2 side you would have to add another subnet for the other EC2 instance and connect them with a load balancer. In case one of the subnets goes down for whatever reason, your site is still up and running.

For this article however, we're going to focus on the minimum setup for development and testing purposes.

**VPC**

To create a VPC we configure our module as follows:

resource "aws\_vpc" "\_" {

cidr\_block = var.vpc\_cidr

enable\_dns\_support = var.enable\_dns\_support

enable\_dns\_hostnames = var.enable\_dns\_hostnames

}

resource "aws\_internet\_gateway" "\_" {

vpc\_id = aws\_vpc.\_.id

}

resource "aws\_route\_table" "\_" {

vpc\_id = aws\_vpc.\_.id

dynamic "route" {

for\_each = var.route

content {

cidr\_block = route.value.cidr\_block

gateway\_id = route.value.gateway\_id

instance\_id = route.value.instance\_id

nat\_gateway\_id = route.value.nat\_gateway\_id

}

}

}

resource "aws\_route\_table\_association" "\_" {

count = length(var.subnet\_ids)

subnet\_id = element(var.subnet\_ids, count.index)

route\_table\_id = aws\_route\_table.\_.id

}

**Please note**; I removed the tag blocks for brevity, but you should tag every resource possible to enable easy cost tracking of your deployments and to be able to find everything should anything go wrong with the tfstate.

The route table is configured to be *associated* with an internet gateway in the aws\_route\_table\_association resource. Any subnet we supply in var.subnet\_ids will have access to the route table configuration and the internet gateway.

I call the VPC module like this:

module "vpc" {

source = "../../modules/vpc"

resource\_tag\_name = var.resource\_tag\_name

namespace = var.namespace

region = var.region

vpc\_cidr = "10.0.0.0/16"

route = [

{

cidr\_block = "0.0.0.0/0"

gateway\_id = module.vpc.gateway\_id

instance\_id = null

nat\_gateway\_id = null

}

]

subnet\_ids = module.subnet\_ec2.ids

}

The vpc\_cidr = "10.0.0.0/16" means we're creating a VPC with 65,536 possible IP addresses. See [here](https://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing) for an explanation on the CIDR notation.

The route table is connected to the EC2 subnet via; subnet\_ids = module.subnet\_ec2.ids. This subnet has full access to the internet via the cidr\_block configuration; "0.0.0.0/0".

**EC2**

To create the EC2 instance, we just need to configure what machine we want and place it in the subnet where our Route Table is present.

In our EC2 module we configure the following:

locals {

resource\_name\_prefix = "${var.namespace}-${var.resource\_tag\_name}"

}

resource "aws\_instance" "\_" {

ami = var.ami

instance\_type = var.instance\_type

user\_data = var.user\_data

subnet\_id = var.subnet\_id

associate\_public\_ip\_address = var.associate\_public\_ip\_address

key\_name = aws\_key\_pair.\_.key\_name

vpc\_security\_group\_ids = var.vpc\_security\_group\_ids

iam\_instance\_profile = var.iam\_instance\_profile

}

resource "aws\_eip" "\_" {

vpc = true

instance = aws\_instance.\_.id

}

resource "tls\_private\_key" "\_" {

algorithm = "RSA"

rsa\_bits = 4096

}

resource "aws\_key\_pair" "\_" {

key\_name = var.key\_name

public\_key = tls\_private\_key.\_.public\_key\_openssh

}

This creates:

1) AWS EC2 instance  
2) With an elastic IP associated with that instance  
3) A public/private key (PEM key) to access the instance via SSH.

Then we can call it:

module "ec2" {

source = "../../modules/ec2"

resource\_tag\_name = var.resource\_tag\_name

namespace = var.namespace

region = var.region

ami = "ami-07ebfd5b3428b6f4d" # Ubuntu Server 18.04 LTS

key\_name = "${local.resource\_name\_prefix}-ec2-key"

instance\_type = var.instance\_type

subnet\_id = module.subnet\_ec2.ids[0]

vpc\_security\_group\_ids = [aws\_security\_group.ec2.id]

vpc\_id = module.vpc.id

}

Four main things we need to supply the EC2 module (among other things):

1) Attach the EC2 instance to the subnet; subnet\_id = module.subnet\_ec2.ids[0],  
2) attaches the security group; vpc\_security\_group\_ids = [aws\_security\_group.ec2.id], a security group acts like a firewall.  
3) Supply it with the VPC that it needs to be deployed in; vpc\_id = module.vpc.id  
4) AMI identifier, [here's](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/finding-an-ami.html) more on how to find Amazon Machine Image (AMI) identifiers.

**EC2 Security group**

So far we have a VPC setup, an EC2 instance and its subnet, and we've configured a reference to the security group the EC2 subnet is using.

A security group acts like a firewall for your subnet, what is allowed to go in ingress and what is allowed to go out egress of your subnet:

resource "aws\_security\_group" "ec2" {

name = "${local.resource\_name\_prefix}-ec2-sg"

description = "EC2 security group (terraform-managed)"

vpc\_id = module.vpc.id

ingress {

from\_port = var.rds\_port

to\_port = var.rds\_port

protocol = "tcp"

description = "MySQL"

cidr\_blocks = local.rds\_cidr\_blocks

}

ingress {

from\_port = 22

to\_port = 22

protocol = "tcp"

description = "Telnet"

cidr\_blocks = ["0.0.0.0/0"]

}

ingress {

from\_port = 80

to\_port = 80

protocol = "tcp"

description = "HTTP"

cidr\_blocks = ["0.0.0.0/0"]

}

ingress {

from\_port = 443

to\_port = 443

protocol = "tcp"

description = "HTTPS"

cidr\_blocks = ["0.0.0.0/0"]

}

# Allow all outbound traffic.

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

}

We allow traffic to come in from ports; **22** (SSH), **80** (HTTP), **443** (HTTPS), and we allow ALL traffic on all ports to go out. If you want to further tighten this down, profile which ports your application uses for outbound traffic to increase security.

For ingress you can further tighten this down by supplying a specific IP address that is allowed to connect on port 22.

**RDS**

We're almost done with the setup, only our database subnet and instance with security group needs to be configured.

locals {

resource\_name\_prefix = "${var.namespace}-${var.resource\_tag\_name}"

}

resource "aws\_db\_subnet\_group" "\_" {

name = "${local.resource\_name\_prefix}-${var.identifier}-subnet-group"

subnet\_ids = var.subnet\_ids

}

resource "aws\_db\_instance" "\_" {

identifier = "${local.resource\_name\_prefix}-${var.identifier}"

allocated\_storage = var.allocated\_storage

backup\_retention\_period = var.backup\_retention\_period

backup\_window = var.backup\_window

maintenance\_window = var.maintenance\_window

db\_subnet\_group\_name = aws\_db\_subnet\_group.\_.id

engine = var.engine

engine\_version = var.engine\_version

instance\_class = var.instance\_class

multi\_az = var.multi\_az

name = var.name

username = var.username

password = var.password

port = var.port

publicly\_accessible = var.publicly\_accessible

storage\_encrypted = var.storage\_encrypted

storage\_type = var.storage\_type

vpc\_security\_group\_ids = ["${aws\_security\_group.\_.id}"]

allow\_major\_version\_upgrade = var.allow\_major\_version\_upgrade

auto\_minor\_version\_upgrade = var.auto\_minor\_version\_upgrade

final\_snapshot\_identifier = var.final\_snapshot\_identifier

snapshot\_identifier = var.snapshot\_identifier

skip\_final\_snapshot = var.skip\_final\_snapshot

performance\_insights\_enabled = var.performance\_insights\_enabled

}

There are a lot of options here, lets grab the tfvars file to see what most of these variables contains:

# RDS

rds\_identifier = "mysql"

rds\_engine = "mysql"

rds\_engine\_version = "8.0.15"

rds\_instance\_class = "db.t2.micro"

rds\_allocated\_storage = 10

rds\_storage\_encrypted = false # not supported for db.t2.micro instance

rds\_name = "" # use empty string to start without a database created

rds\_username = "admin" # rds\_password is generated

rds\_port = 3306

rds\_maintenance\_window = "Mon:00:00-Mon:03:00"

rds\_backup\_window = "10:46-11:16"

rds\_backup\_retention\_period = 1

rds\_publicly\_accessible = false

rds\_final\_snapshot\_identifier = "prod-trademerch-website-db-snapshot" # name of the final snapshot after deletion

rds\_snapshot\_identifier = null # used to recover from a snapshot

rds\_performance\_insights\_enabled = true

A few notes on the configuration I used here;

1) **Instance sizing and encryption:** for production make sure you use an instance size that is larger than a db.t2.micro such that you can use encryption on the storage layer.  
2) **Maintenance window:** This day and time setting is used for patching of your instance.  
3) **Public access:** Make sure to set public access off for obvious reasons, but this should already be the case anyway if your instance is hosted in a private subnet.  
4) **Backups:** Two things regarding backups:  
4.1) Providing the final snapshot identifier is useful when destroying the environment, it will automatically create a snapshot with the given name. If you do no supply this variable, you wont be able to remove the RDS instance with the terraform destroy command and you'll have to do this manually(!).  
4.2) RDS supports automated backups, make sure to set the retention period (in days) correctly.  
5) **Query tracing:** To enable in depth tracing of your queries and performance statistics, set performance\_insights\_enabled to true. This is very useful in analyzing slow queries and, generally, query performance.  
6) **Database password:** The password for the database is generated, this can be done with this resource:

resource "random\_string" "password" {

length = 16

special = false

}

**RDS Security group**

Finally, we need to supply the security group configuration for RDS such that EC2 can communicate with our Database.

resource "aws\_security\_group" "\_" {

name = "${local.resource\_name\_prefix}-rds-sg"

description = "RDS (terraform-managed)"

vpc\_id = var.rds\_vpc\_id

# Only MySQL in

ingress {

from\_port = var.port

to\_port = var.port

protocol = "tcp"

cidr\_blocks = var.sg\_ingress\_cidr\_block

}

# Allow all outbound traffic.

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = var.sg\_egress\_cidr\_block

}

}

The above allows ingress from port 3306 and egress everything.

**Conclusion**

I hope this guide has been useful, please leave a comment below to let me know what you liked, did not like, suggestions and so on.

Next week I'll cover (Open)API security with configuration recommendations, and an AWS API firewall solution, [AWS WAF](https://aws.amazon.com/waf/).

Thanks for reading!

**Top comments (6)**

Subscribe

Top of Form





Bottom of Form

[amaurybsouza profile image](https://dev.to/amaurybsouza)

Amaury Borges Souza

•[Apr 20 '20](https://dev.to/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij#comment-o1cp)

Hi, I thinked your article very nice. I work with Terraform and I'm with a doubt.

How I define a database password using variables with Terraform?

Many thanks.

3 likesLike[Reply](https://dev.to/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij#/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij/comments/new/o1cp)

[rolfstreefkerk profile image](https://dev.to/rolfstreefkerk)

Rolf Streefkerk

•[Apr 21 '20](https://dev.to/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij#comment-o21n)

If you do not want to generate it with the random\_string resource, you can just supply it in your environment configuration file (tfvars file), but of course the caveat here is that if you check in this file into github it's visible in plaintext. Where as the Terraform generated state file can be stored in an encrypted S3 bucket.

The other option is to enable IAM role access, which is the safest way actually to set up authorization over using an explicitly set password as I did. Check this article how to do that:  
[aws.amazon.com/premiumsupport/know...](https://aws.amazon.com/premiumsupport/knowledge-center/users-connect-rds-iam/)

I hope that helps

3 likesLike[Reply](https://dev.to/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij#/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij/comments/new/o21n)

[rolfstreefkerk profile image](https://dev.to/rolfstreefkerk)

Rolf Streefkerk

•[Apr 24 '20](https://dev.to/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij#comment-o5ch)

There's another option, using Secrets Manager. See this article how that's done.  
[github.com/aws-samples/aws-serverl...](https://github.com/aws-samples/aws-serverless-security-workshop/tree/master/docs/02-add-secrets-manager)

In terms of security, I'd rate it:  
1) IAM  
2) Secrets Manager  
3) Terraform state storage on encrypted S3  
4) Input at Terraform deployment

3 likesLike[Reply](https://dev.to/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij#/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij/comments/new/o5ch)

[charlesdlandau profile image](https://dev.to/charlesdlandau)

Charles Landau

•[Mar 8 '20](https://dev.to/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij#comment-mc3f)

Love this! I've been enjoying this module which scratches a similar itch - you may find it useful!

[github.com/terraform-aws-modules/t...](https://github.com/terraform-aws-modules/terraform-aws-vpc)

2 likesLike[Reply](https://dev.to/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij#/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij/comments/new/mc3f)

[rolfstreefkerk profile image](https://dev.to/rolfstreefkerk)

Rolf Streefkerk

•[Mar 9 '20](https://dev.to/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij#comment-md2k)

this module is very complete, nice catch.

1 likeLike[Reply](https://dev.to/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij#/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij/comments/new/md2k)

[amiller68 profile image](https://dev.to/amiller68)

Al Miller

•[Sep 21 '22](https://dev.to/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij#comment-21nao)

Do you have a link to a Repo for this? I'm new to terraform and trying to understand how the modules are structured.

1 likeLike[Reply](https://dev.to/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij#/rolfstreefkerk/how-to-setup-a-basic-vpc-with-ec2-and-rds-using-terraform-3jij/comments/new/21nao)

[Code of Conduct](https://dev.to/code-of-conduct) • [Report abuse](https://dev.to/report-abuse)

DEV Community

**Trending in Database**

The database community is currently engaged in discussions around NoSQL databases like **MongoDB** and **Firestore**, the importance of choosing the right database for application performance, and the impact of index optimization on query performance. There's also interest in creating data maps with **Matplotlib** and understanding multi-tenant SaaS architecture with **Entity Framework**.

[[](https://dev.to/kalashin1)](https://dev.to/kalashin1)

**[How to Choose the Right Document-Oriented NoSQL Database for Your Application](https://dev.to/kalashin1/how-to-choose-the-right-document-oriented-nosql-database-for-your-application-3hac)**

**[Kinanee Samson ・ Sep 4](https://dev.to/kalashin1/how-to-choose-the-right-document-oriented-nosql-database-for-your-application-3hac)**

[#database #mongodb #fauna #nosql](https://dev.to/kalashin1/how-to-choose-the-right-document-oriented-nosql-database-for-your-application-3hac)

[[](https://dev.to/taquiimam14)](https://dev.to/taquiimam14)

**[Choosing the Right Database for Your Application](https://dev.to/taquiimam14/choosing-the-right-database-for-your-application-5ff3)**

**[Taqui ・ Sep 11](https://dev.to/taquiimam14/choosing-the-right-database-for-your-application-5ff3)**

[#webdev #javascript #programming #database](https://dev.to/taquiimam14/choosing-the-right-database-for-your-application-5ff3)

[[](https://dev.to/oscarleo)](https://dev.to/oscarleo)

**[How to Create Data Maps of the United States With Matplotlib](https://dev.to/oscarleo/how-to-create-data-maps-of-the-united-states-with-matplotlib-p9i)**

**[Oscar Leo ・ Sep 5](https://dev.to/oscarleo/how-to-create-data-maps-of-the-united-states-with-matplotlib-p9i)**

[#tutorial #python #database #datavisualization](https://dev.to/oscarleo/how-to-create-data-maps-of-the-united-states-with-matplotlib-p9i)

[[](https://dev.to/developerpartners)](https://dev.to/developerpartners)

**[Multi-Tenant SaaS Architecture with Entity Framework](https://dev.to/developerpartners/multi-tenant-saas-architecture-with-entity-framework-omc)**

**[Developer Partners ・ Sep 7](https://dev.to/developerpartners/multi-tenant-saas-architecture-with-entity-framework-omc)**

[#dotnet #sql #csharp #database](https://dev.to/developerpartners/multi-tenant-saas-architecture-with-entity-framework-omc)

[[](https://dev.to/nightbird07)](https://dev.to/nightbird07)

**[Index Optimization in Database Query Performance: Guidelines and Real-world Examples](https://dev.to/nightbird07/index-optimization-in-database-query-performance-guidelines-and-real-world-examples-4f80)**

**[NightBird07 ・ Sep 11](https://dev.to/nightbird07/index-optimization-in-database-query-performance-guidelines-and-real-world-examples-4f80)**

[#postgres #tutorial #database #beginners](https://dev.to/nightbird07/index-optimization-in-database-query-performance-guidelines-and-real-world-examples-4f80)

**Read next**

[](https://dev.to/rahulkarda/top-aws-serverless-plugins-you-need-to-know-55dh)

**[Top AWS Serverless Plugins You Need to Know](https://dev.to/rahulkarda/top-aws-serverless-plugins-you-need-to-know-55dh)**

[Rahul Karda - Aug 24](https://dev.to/rahulkarda/top-aws-serverless-plugins-you-need-to-know-55dh)

[](https://dev.to/newrelic/new-relic-grok-the-industrys-first-genai-assistant-for-observability-is-in-preview-1ih8)

**[New Relic Grok, the industry’s first GenAI assistant for observability, is in preview](https://dev.to/newrelic/new-relic-grok-the-industrys-first-genai-assistant-for-observability-is-in-preview-1ih8)**

[alvaradodaniel3 - Aug 23](https://dev.to/newrelic/new-relic-grok-the-industrys-first-genai-assistant-for-observability-is-in-preview-1ih8)

[](https://dev.to/memgraph/in-memory-vs-disk-based-databases-why-do-you-need-a-larger-than-memory-architecture-37p)

**[In-memory vs. disk-based databases: Why do you need a larger than memory architecture?](https://dev.to/memgraph/in-memory-vs-disk-based-databases-why-do-you-need-a-larger-than-memory-architecture-37p)**

[Memgraph - Sep 5](https://dev.to/memgraph/in-memory-vs-disk-based-databases-why-do-you-need-a-larger-than-memory-architecture-37p)

[](https://dev.to/devopsbridge/how-can-devops-consulting-services-streamline-your-development-and-operations-2jgd)

**[How can DevOps Consulting Services Streamline Your Development and Operations?](https://dev.to/devopsbridge/how-can-devops-consulting-services-streamline-your-development-and-operations-2jgd)**

[Saif - Aug 23](https://dev.to/devopsbridge/how-can-devops-consulting-services-streamline-your-development-and-operations-2jgd)

[Rolf Streefkerk](https://dev.to/rolfstreefkerk)

Follow

AWS Solution Architect Professional interested in product development

* LOCATION

Bangkok

* EDUCATION

MSc Software Engineering

* WORK

AWS Solution Architect at Multithread Labs

* JOINED

Jan 8, 2020

**More from**[**Rolf Streefkerk**](https://dev.to/rolfstreefkerk)

[How to setup a Serverless application with AWS SAM and Terraform](https://dev.to/rolfstreefkerk/how-to-setup-a-serverless-application-with-aws-sam-and-terraform-33m9)

[#serverless #terraform #devops #aws](https://dev.to/rolfstreefkerk/how-to-setup-a-serverless-application-with-aws-sam-and-terraform-33m9)

[How to protect Serverless (Open)API's?](https://dev.to/rolfstreefkerk/how-to-protect-serverless-open-api-s-5eem)

[#aws #security #devops #webdev](https://dev.to/rolfstreefkerk/how-to-protect-serverless-open-api-s-5eem)

[Serverless CI/CD for AWS Lambda](https://dev.to/rolfstreefkerk/serverless-ci-cd-for-aws-lambda-3hpg)

[#serverless #devops #aws](https://dev.to/rolfstreefkerk/serverless-ci-cd-for-aws-lambda-3hpg)

DEV Community

**🌚 Life is too short to browse without**[**dark mode**](https://dev.to/settings/customization)

You can customize your theme, font, and more [when you are signed in](https://dev.to/enter?state=new-user).

[DEV Community](https://dev.to/) — A constructive and inclusive social network for software developers. With you every step of your journey.

* [Home](https://dev.to/)
* [Podcasts](https://dev.to/pod)
* [Videos](https://dev.to/videos)
* [Tags](https://dev.to/tags)
* [FAQ](https://dev.to/faq)
* [Forem Shop](https://shop.forem.com/)
* [Advertise on DEV](https://dev.to/dev-advertising-options)
* [About](https://dev.to/about)
* [Contact](https://dev.to/contact)
* [Guides](https://dev.to/guides)
* [Software comparisons](https://dev.to/software-comparisons)
* [Code of Conduct](https://dev.to/code-of-conduct)
* [Privacy Policy](https://dev.to/privacy)
* [Terms of use](https://dev.to/terms)

Built on [Forem](https://www.forem.com/" \t "_blank) — the [open source](https://dev.to/t/opensource) software that powers [DEV](https://dev.to/) and other inclusive communities.

Made with love and [Ruby on Rails](https://dev.to/t/rails). DEV Community © 2016 - 2023.