Even the best database depends on running on solid hardware. And the cloud gives you flexibility at a reasonable price. We have already explained how to deploy GraphDB in a generic environment and on Azure. Ontotext does not stop here, of course. We collaborate with AWS as well. This post will guide you through deploying GraphDB in your AWS account using our Terraform scripts.

The scripts

The Terraform scripts are located in the terraform-aws-graphdb repository of the Ontotext-AD GitHub organization. The scripts deploy the cluster using pre-built AMIs with the latest GraphDB version. Building the images with Packer is part of the GraphDB release process. The implementation can also be found in the packer-aws-graphdb repository.

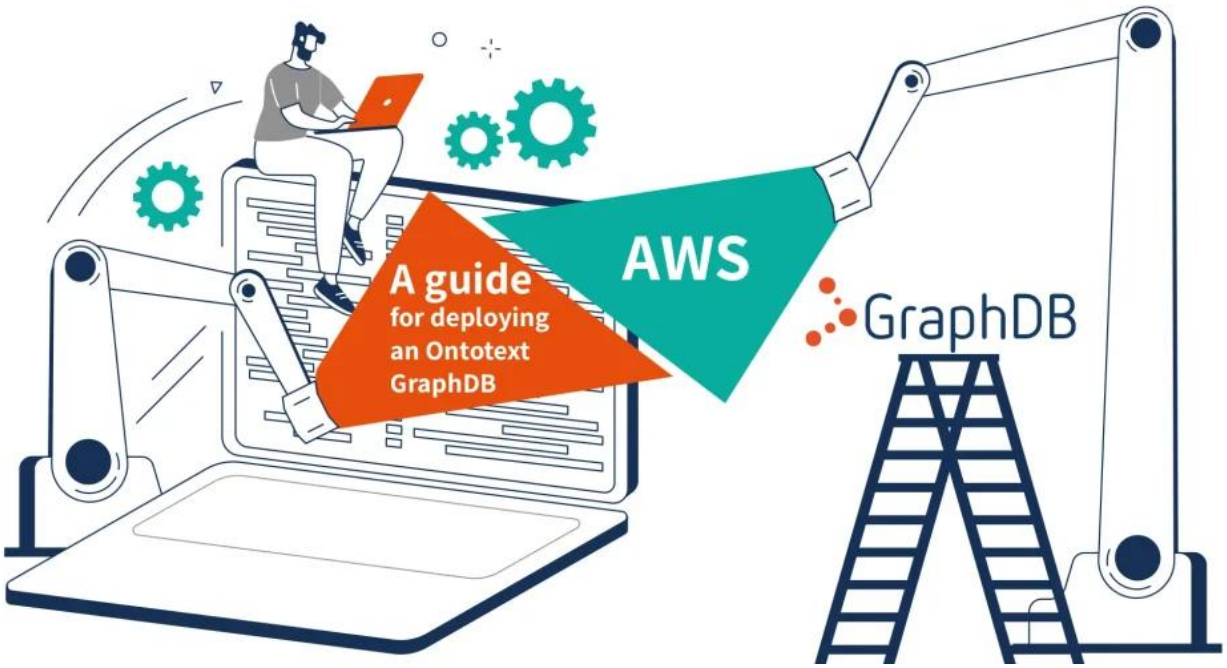
The Terraform scripts bring up all the required AWS components and run additional bash scripts that set up the GraphDB cluster, security, backups, and related infrastructure. The default cluster size is three nodes, and this can be configured based on your license. How to deploy GraphDB in AWS in GraphDB's documentation describes the architecture in more detail.

The Terraform script also sets up important features critical for a live production system. Those include:

- **SSL Termination** — decrypting encrypted data using the AWS load balancer.
- **Backup** — the data is crucial for any enterprise and a regular backup is a must-have.
- **Monitoring and alerting** — having a centralized monitoring and alerting system lets you track the cluster's health.

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

ontotext



Deploying a 3-node cluster

You will need the following prerequisites before proceeding with the deployment:

- **An AWS account** with sufficient permissions to deploy the different resources
- **Knowledge of Terraform** (although not strictly necessary, this will help when troubleshooting potential problems)
- **AWS CLI** installed and configured
- **GraphDB license** (the 3-node cluster requires a GraphDB Enterprise Edition License, which can be a trial license)
- **An accepted BYOL offer** for AWS, so the AMI image is found

The first step of setting up a GraphDB cluster on AWS is to clone or download the repository with the Terraform scripts or use the repository as a sub-module from the Terraform registry. Make sure that you clone the latest release. Before executing the scripts it's good to review the `README.md` file, which gives additional information about deploying and configuring the cluster.

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

Below is a `.tfvars` file containing example values for some of the variables that need to be passed to the Terraform to deploy the cluster correctly. To use it, create a file called `terraform.tfvars` to store these values:

```
# This is the resource name prefix for all the resources
# that will be created.
resource_name_prefix = "gdb-cluster"
# The regions where we will deploy.
aws_region = "us-east-1"
# Custom tags in case you need them. We recommend using those
common_tags = {
Environment: "integration"
CostCenter: "organisation"
}
# Prevent resources that support purge protection from being deleted.
prevent_resource_deletion = true
# The path to the GraphDB license.
# This should be a file that you have received from Ontotext.
graphdb_license_path = "/Users/user/licenses/graphdb.license"
# We recommend setting up an admin password here, you can always
change it later
graphdb_admin_password = "some-password"
# The version of GraphDB. If not set it will always deploy
# the pre-built VM with the latest version.
graphdb_version = "10.7.3"
# The type of the machines that will be used for the deployment. Think
of your requirements and chose carefully
ec2_instance_type = "m5.xlarge"
allowed_inbound_cidrs_lb = ["0.0.0.0/0"]
# Path to certificate for TLS. It should be stored in ACM.
lb_tls_certificate_arn = "arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"
# Optional key name, used for SSH. It should be created in EC2.
ec2_key_name = "SSH-test"
```

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

ontotext

```
# Optional list of CIDR blocks to permit for SSH to nodes.
# You would also need to open port 22 on the EC2 firewall.
allowed_inbound_cidrs_ssh = ["0.0.0.0/0"]
# For the sake of a test deployment we turn off the monitoring.
# By default this will also deploy the monitoring and alerting.
deploy_monitoring = false
```

Once we have the file with all the input parameters, we are ready to initialize it with the following command, which will look for the terraform.tfvars file in the directory where you execute this command:

```
% terraform init
```

Before deploying, make sure to inspect the plan output. This lets you review the resources that Terraform will create. You can inspect the plan output by running:

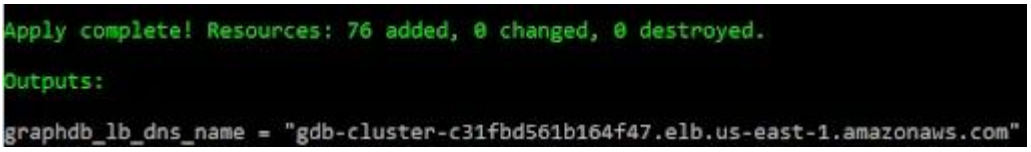
```
% terraform plan
```

Next, deploy with:

```
% terraform apply
```

The deployment process takes around 15 minutes. Sometimes it can take more, depending on AWS. After this, you should see console output like the following:

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com



The graphdb_lb_dns_name points to the address of your GraphDB instance. Keep in mind that when Terraform has finished and you have the address of your new GraphDB instance, it will take some time until the cluster is fully set up. This is because the scripts need to set up the cluster, backups, security, and more. The Workbench will not open until all the scripts are done. You can follow this progress in the cloud-init log file.

Some organizations will not want a public GraphDB address, so the Terraform script also supports a private link.

Note that if you had a failed deployment of GraphDB with the terraform script, you should manually delete the EBS volumes that were created to clean up the results of the automated initialization scripts.

Operating GraphDB

The GraphDB nodes are deployed in AWS EC2 instances, which are a part of an Auto Scaling Group. By default, public SSH access to these VMs is blocked. If you need to SSH into any of them, you should configure your key and only allow inbound SSH access from your address or addresses, using `allowed_inbound_cidrs_ssh`

Once you are logged in, it might be useful to know what is where. Each VM contains a GraphDB instance and an external GraphDB cluster proxy where both are started as a `system.d` service. The folders where their files are stored are:

- `/var/opt/graphdb/node` — for the GraphDB instance
- `/var/opt/graphdb/cluster-proxy` — for the cluster proxy

Those might be needed if you need to change configurations or to examine logs when you don’t have the monitoring enabled.

The backups are another important part of the deployment. They are created and uploaded

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

ontotext

in AWS S3 storage daily at midnight, although you can configure this and many more options via Terraform or change them manually. The duration of the backup process will depend on your data size and machines. The following are some variables that you can use to configure the backups. For more options, please refer to the `variables.tf` file in the terraform-aws-graphdb GitHub repository.

- You can use the `backup_schedule` to change the cron expression that defines when the backup is executed
- `backup_retention_count` — defines how many backups to keep, with the default being 7

Monitoring

The monitoring module is disabled in the `.tfvars` example file but if you enable it, you can find all the metrics and alerts in the interactive AWS CloudWatch tool. The setup and configuration have been integrated into the VM image and Terraform scripts.

After a successful deployment in AWS via the AMI and scripts you will notice the following resources:

- **Cloud watch dashboard**: a dashboard for logging data from CloudWatch Agent and other AWS services such as Route 53 checks
- **A SNS topic**, <resource_name_prefix>-graphdb-notifications: used for notifications (push|e-mail|sms) to `users/roles`
- **Additional alerts**: We've set up several alerts based on ingested logs, signals, and availability tests:
  - Low memory alert: will send a notification if the used memory exceeds the configured percentage
  - Availability alert: will send a notification if the availability drops beneath 100
  - Low disk space alert: will send a notification if the "low disk space" message is detected in the ingested logs
  - Replication in progress alert: will send a notification if a request for snapshot replication is detected in the ingested logs

Cloud watch also includes predefined workbooks that cover detailed analytics for Availability,

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

Failures, Performance, and more, which can be further modified and saved.

What’s next?

This is a rough guideline on how to deploy a GraphDB high-availability cluster. More technical details can be found in the GraphDB documentation and you can see all parameters in the GitHub repository. Also, we are active on GitHub and are interested in suggestions for improvements or new features.

If you don’t want to tackle deploying GraphDB yourself, contact Ontotext’s sales team and ask about the SaaS version. You can also buy it directly from the AWS Marketplace.

Maximize the power of your enterprise data with GraphDB on AWS!

Give It a Try

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com