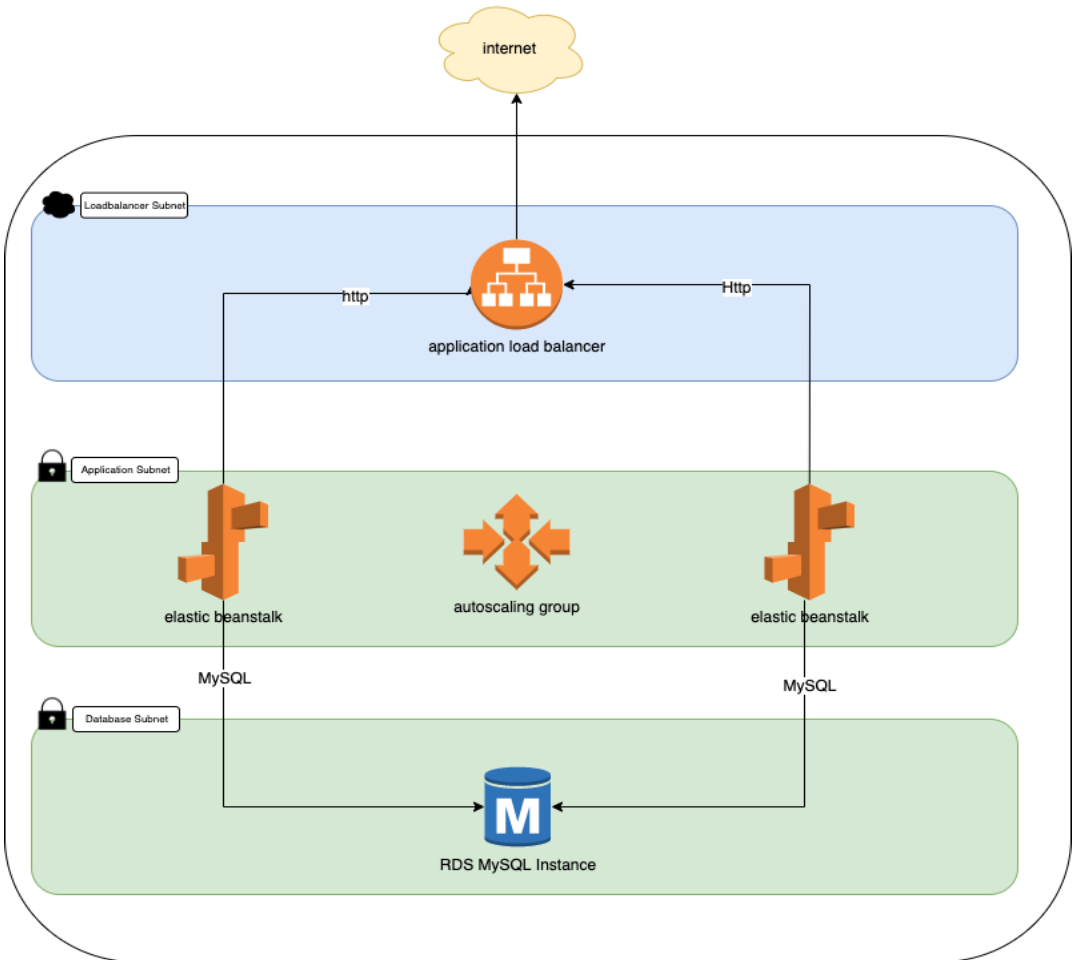
Business Requirements

- The Application must serve variable amount of traffic. Most users are active during business hours. During big events and conferences the traffic could be 4 times more than typical.
- The Customer takes guarantee to preserve your notes up to 3 years and recover it if needed.
- The Customer ensures continuity in service in case of datacenter failures.
- The Service must be capable of being migrated to any regions supported by the cloud provider in case of emergency.
- The Customer is planning to have more than 100 developers to work in this project who want to roll out multiple deployments a day without interruption / downtime.
- The Customer wants to provision separated environments to support their development process for development, testing, production in the near future.
- The Customer wants to see relevant metrics and logs from the infrastructure for quality assurance and security purposes.

Architecture Draft



Core components:

- AWS Elastic Beanstalk (node.js) in autoscaling group (min=1 max=4)
- RDS Database (MySQL) in Multi-AZ

- One VPC
- 2 x Public Subnets for Load Balancer
- 2 x Private Subnets (with NATGW) for APP
- 2 x Private Subnets (without Internet) for DB

Application redesign:

- SQLite database has been replaced by MySQL. One of the project requirements was "Application must serve variable amount of traffic." It means that it needs to be more scalable and ready for the multiple instances of the same application accessing a database at the same time. One of official SQLite recommendation is migrate to the client/server database engine because they usually support a higher level of concurrency and offer better performance: https://www.sqlite.org/faq.html#q5

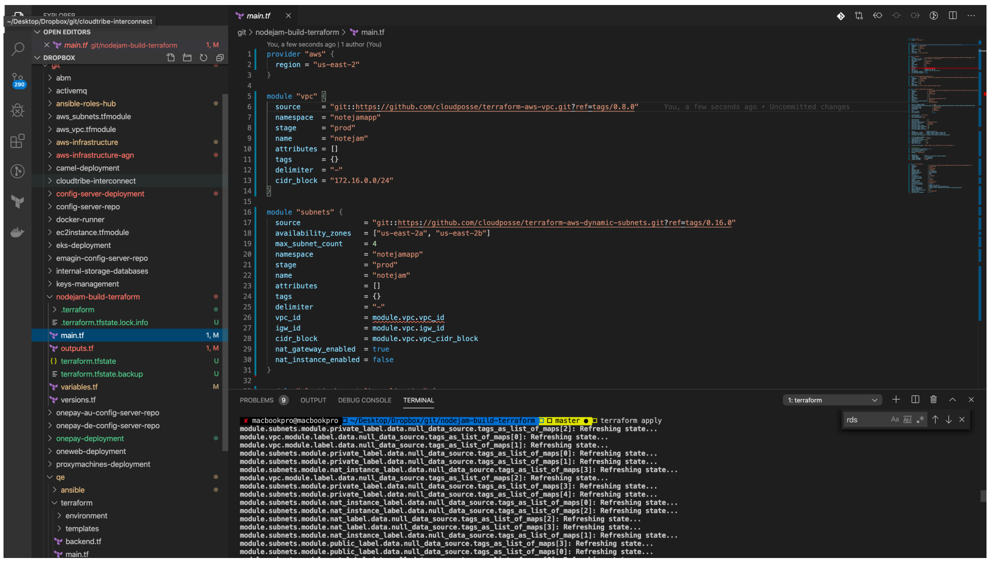
- Database initialization script (db.js) has been rewritten to create MySQL schema.

- package.json has been modified to initiate database schema before start application

- bcrypt library has been replaced by bcryptjs: This change was due to problems with building the elastic beanstalk application using the "bcrypt" library. Both libraries are functionally the same, bcryptjs is a pure js module.

Cloud Environment:

- Environment has been prepared to build automatically using Terraform (0.12.x) scripts. Terraform is using Cloud Posse modules, available on Apache2 license: http://github.com/cloudposse/



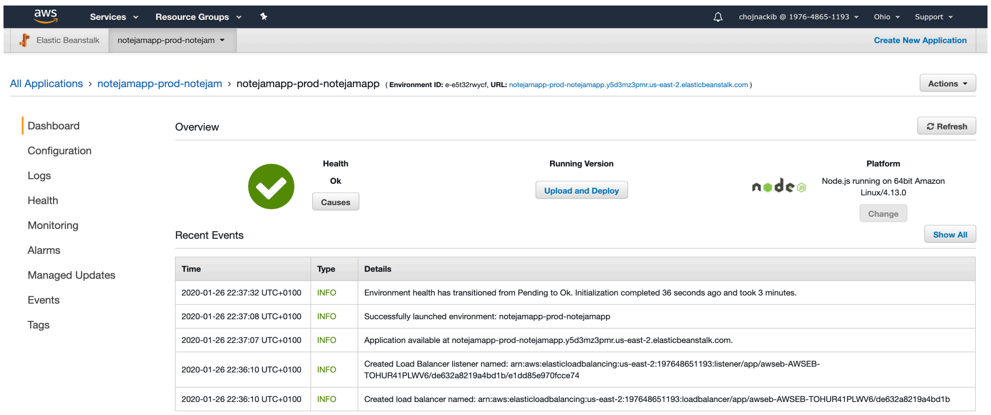
```
provider "aws" {
  region = "us-east-2"
}

module "vpc" {
  source     = "git::https://github.com/cloudposse/terraform-aws-vpc.git?ref=tags/0.8.0"      You, a few seconds ago • Uncommitted changes
  namespace  = "notejamapp"
  stage      = "prod"
  name       = "notejam"
  attributes = []
  tags       = {}
  delimiter  = "-"
  cidr_block = "172.16.0.0/24"
}

module "subnets" {
  source               = "git::https://github.com/cloudposse/terraform-aws-dynamic-subnets.git?ref=tags/0.16.0"
  availability_zones   = ["us-east-2a", "us-east-2b"]
  max_subnet_count     = 4
  namespace            = "notejamapp"
  stage                = "prod"
  name                 = "notejam"
  attributes           = []
  tags                 = {}
  delimiter            = "-"
  vpc_id               = module.vpc.vpc_id
  igw_id               = module.vpc.igw_id
  cidr_block           = module.vpc.vpc_cidr_block
  nat_gateway_enabled  = true
  nat_instance_enabled = false
}
```

```
✘ macbookpro@macbookpro  ~/Desktop/Dropbox/git/nodejam-build-terraform  master ●  terraform apply
module.subnets.module.private_label.data.null_data_source.tags_as_list_of_maps[2]: Refreshing state...
module.vpc.module.label.data.null_data_source.tags_as_list_of_maps[0]: Refreshing state...
module.vpc.module.label.data.null_data_source.tags_as_list_of_maps[1]: Refreshing state...
module.subnets.module.private_label.data.null_data_source.tags_as_list_of_maps[0]: Refreshing state...
module.subnets.module.private_label.data.null_data_source.tags_as_list_of_maps[1]: Refreshing state...
module.subnets.module.nat_instance_label.data.null_data_source.tags_as_list_of_maps[3]: Refreshing state...
module.vpc.module.label.data.null_data_source.tags_as_list_of_maps[2]: Refreshing state...
module.subnets.module.private_label.data.null_data_source.tags_as_list_of_maps[3]: Refreshing state...
module.subnets.module.private_label.data.null_data_source.tags_as_list_of_maps[4]: Refreshing state...
module.subnets.module.nat_instance_label.data.null_data_source.tags_as_list_of_maps[0]: Refreshing state...
module.subnets.module.nat_instance_label.data.null_data_source.tags_as_list_of_maps[2]: Refreshing state...
module.subnets.module.nat_label.data.null_data_source.tags_as_list_of_maps[2]: Refreshing state...
module.subnets.module.nat_label.data.null_data_source.tags_as_list_of_maps[3]: Refreshing state...
module.subnets.module.nat_instance_label.data.null_data_source.tags_as_list_of_maps[1]: Refreshing state...
module.subnets.module.public_label.data.null_data_source.tags_as_list_of_maps[3]: Refreshing state...
module.subnets.module.public_label.data.null_data_source.tags_as_list_of_maps[0]: Refreshing state...
```

- Application can be deployed From Elastic Beanstalk dashboard or with usage of Elastic Beanstalk CLI by dev/devops team members. (this can be automated with some CI/CD tools like Jenkins/Gitlab etc)

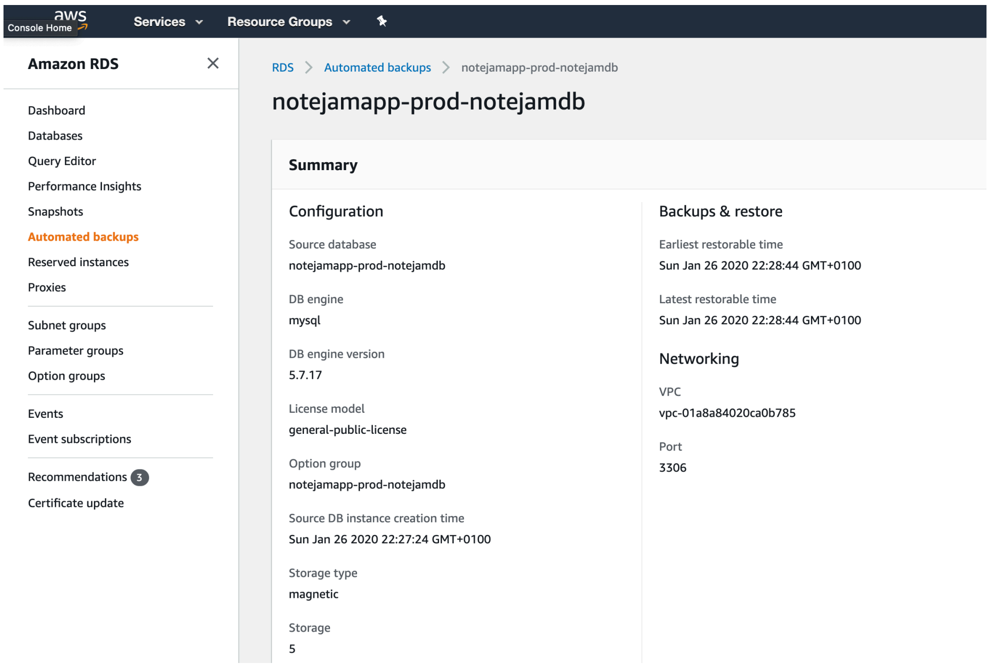


All Applications › notejamapp-prod-notejam › notejamapp-prod-notejamapp (**Environment ID:** e-e5t32rwycf, **URL:** notejamapp-prod-notejamapp.y5d3mz3pmr.us-east-2.elasticbeanstalk.com)

Platform: Node.js running on 64bit Amazon Linux/4.13.0

Time	Type	Details
2020-01-26 22:37:32 UTC+0100	INFO	Environment health has transitioned from Pending to Ok. Initialization completed 36 seconds ago and took 3 minutes.
2020-01-26 22:37:08 UTC+0100	INFO	Successfully launched environment: notejamapp-prod-notejamapp
2020-01-26 22:37:07 UTC+0100	INFO	Application available at notejamapp-prod-notejamapp.y5d3mz3pmr.us-east-2.elasticbeanstalk.com.
2020-01-26 22:36:10 UTC+0100	INFO	Created Load Balancer listener named: arn:aws:elasticloadbalancing:us-east-2:197648651193:listener/app/awseb-AWSEB-TOHUR41PLWV6/de632a8219a4bd1b/e1dd85e970fcce74
2020-01-26 22:36:10 UTC+0100	INFO	Created load balancer named: arn:aws:elasticloadbalancing:us-east-2:197648651193:loadbalancer/app/awseb-AWSEB-TOHUR41PLWV6/de632a8219a4bd1b

- Database setup to be daily snapshotted. There is possible to recover data from snapshot.



notejamapp-prod-notejamdb

Summary

Configuration

Source database
notejamapp-prod-notejamdb

DB engine
mysql

DB engine version
5.7.17

License model
general-public-license

Option group
notejamapp-prod-notejamdb

Source DB instance creation time
Sun Jan 26 2020 22:27:24 GMT+0100

Storage type
magnetic

Storage
5

Backups & restore

Earliest restorable time
Sun Jan 26 2020 22:28:44 GMT+0100

Latest restorable time
Sun Jan 26 2020 22:28:44 GMT+0100

Networking

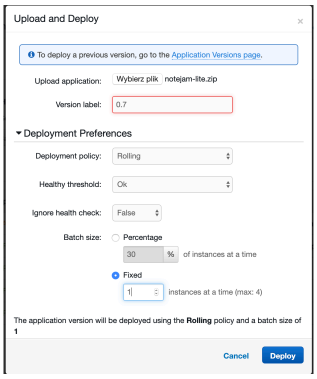
VPC
vpc-01a8a84020ca0b785

Port
3306

- Application is working in Multi-AZ. In case of failure of one datacenter application will be able to work from second AZ.

- There is possibility to deploy all environments in different Region using attached Terraform scripts

- Rolling deployment has been setup "by default". All deployments should be procced without any downtime.



To deploy a previous version, go to the Application Versions page.

Upload application: notejam-lite.zip

Version label: 0.7

Deployment Preferences

Deployment policy: Rolling

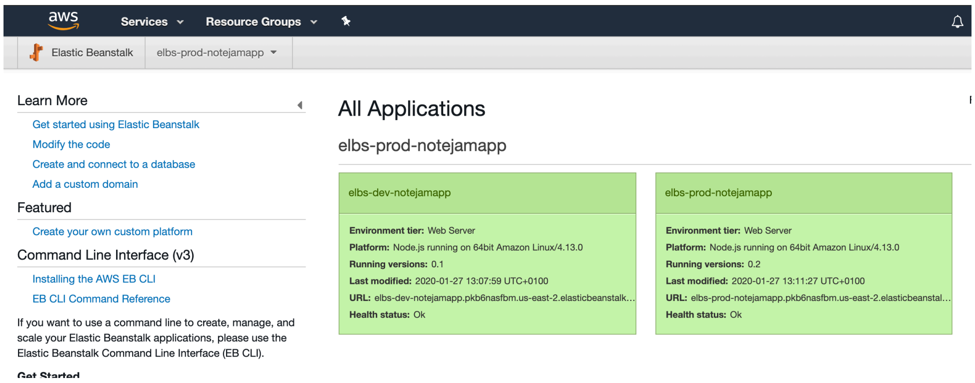
Healthy threshold: Ok

Ignore health check: False

Batch size: Percentage 30 % of instances at a time [✓ Fixed] 1 instances at a time (max: 4)

The application version will be deployed using the **Rolling** policy and a batch size of **1**

- Second "DEV" Environment for development porpoise has been created.


aws
Services
Resource Groups
Elastic Beanstalk
elbs-prod-notejamapp
Learn More
Get started using Elastic Beanstalk
Modify the code
Create and connect to a database
Add a custom domain
Featured
Create your own custom platform
Command Line Interface (v3)
Installing the AWS EB CLI
EB CLI Command Reference
Get Started

If you want to use a command line to create, manage, and scale your Elastic Beanstalk applications, please use the Elastic Beanstalk Command Line Interface (EB CLI).

All Applications

elbs-prod-notejamapp

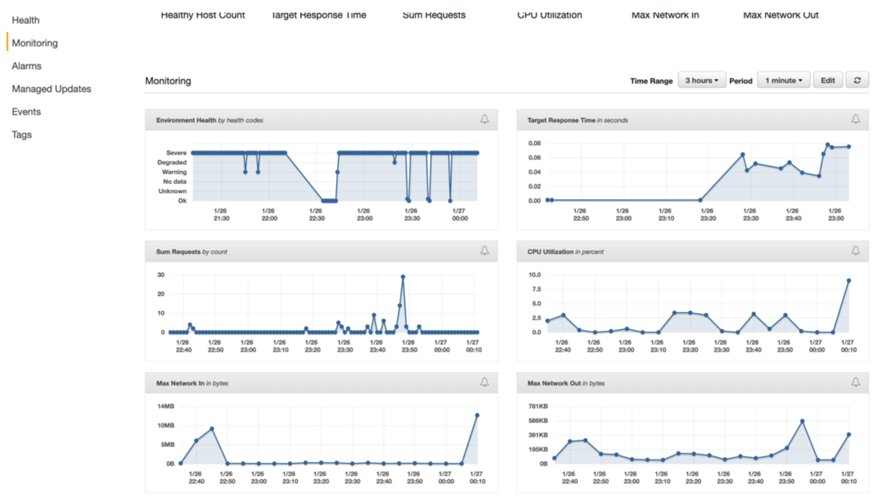
elbs-dev-notejamapp

Environment tier: Web Server
Platform: Node.js running on 64bit Amazon Linux/4.13.0
Running versions: 0.1
Last modified: 2020-01-27 13:07:59 UTC+0100
URL: elbs-dev-notejamapp.pkb6nasfbm.us-east-2.elasticbeanstalk...
Health status: Ok

elbs-prod-notejamapp

Environment tier: Web Server
Platform: Node.js running on 64bit Amazon Linux/4.13.0
Running versions: 0.2
Last modified: 2020-01-27 13:11:27 UTC+0100
URL: elbs-prod-notejamapp.pkb6nasfbm.us-east-2.elasticbeanstal...
Health status: Ok

- All metrics are available from Elastic beanstalk metrics sheet.



Terraform Deployment:

- Please install and configure aws cli on your workstation to be able to connect to your AWS Account
- Please install terraform (recommended version 0.12.19 or higher)
- Check variables.tf to setup:
  - AWS Region
  - Network Range
  - DEV and Production Database Password
- Please provide commands:
  - “terraform init” to initiate terraform
  - “terraform plan” to check what exactly will be build
  - “terraform apply” to build infrastructure

Application Deployment:

- On Elastic Beanstalk Environment menu, please select “Upload & Deploy” button
- Please upload zipped latest source files of your application, and chose “Deployment Preferences”. There are setup Rolling deployment by default, but you can modify deployment strategy as needed.

TODO:

- There is possible to implement CI/CD pipelines for application deployment using “AWS Code”. Application can be kept in git repository on “AWS Code Commit” or external git services. There is possible to create automatic deployment action, using git as a trigger to automatically deliver new version of application.
- I case that in the future developers might need to use microservices as a more sophisticated tool, we could use Fargate instead of Beanstalk in this architecture. This would ease CI/CD part as you can force Fargate Cluster to deploy new tasks automatically from terraform. New task definition would be linked with given docker image from ECR based on tag provided as terraform variable.