envivo

Infrastructure, Architecture and IT Security

Concept / Description / Overview

Version 2
07. Januar 2020

Table of Content

1 Infrastructure

2 Architecture

2.1 Production Environment

2.2 Development / Staging Environment

2.3 Interfaces

3 Disaster Recovery Plan

4 IT Security

4.1 API

4.2 Frontend / Client Portal

4.3 Backoffice

4.4 Static Assets

4.5 User and Client Data

4.6 Solution Platform

4.7 Infrastructure

4.8 CI – CD

1. Infrastructure

Solution is deployed on AWS infrastructure.

AWS solutions for hosting include:

- Route53
- VPC
- EC2
- ECS
- ALB
- S3
- CloudFront
- RDS
- ElastiCache
- Autoscaling
- Security Groups

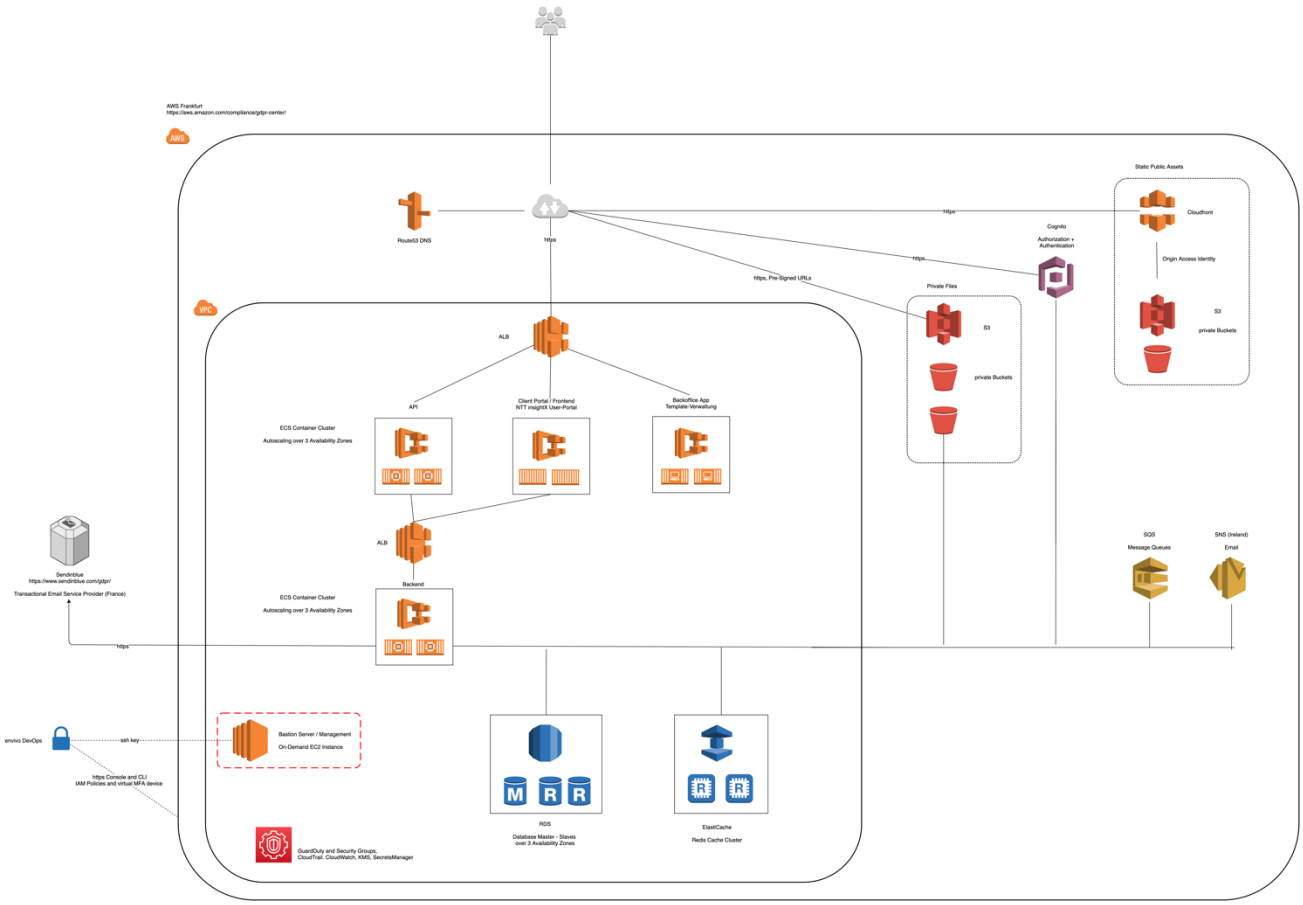
Other AWS solutions used include:

- AWS IAM
- AWS SQS
- AWS SES
- AWS SNS
- AWS GuardDuty
- AWS Security Hub
- AWS CloudWatch
- AWS Key Management Service
- AWS Secrets Manager

Solution is deployed across the EMEA Regions in DE – Frankfurt with multiple Availability Zones.

2. Architecture

2.1 Production Environment



2.2 Development / Staging Environment

Development and Staging Environments are separated, having the same Architecture as Production, but with reduced scaling (ECS / RDS / ElastiCache Cluster scaled only over 1 Availability Zone).

Deployment strategy is described in 4.8

2.3 Interfaces

- AWS APIs via SDK (https)
- Sendinblue API via https
- MySQL via tcp with SSL
- Redis via tcp with SSL

3. Disaster Recovery Plan

- RDS - Multi AZ, Replication mode, Hourly Snapshots, Daily Backup scheduled in AWS. Retention time 30 days.
- AWS CloudWatch to keep track for alarms
- AWS Services are taken across multiple AZs if possible
- Load balancers, autoscaling, ECS, and health monitoring setup
- S3 versioned Objects to mitigate accidental deletions
- Manual steps are specified to restore in a disaster scenario
- RTO is < 1 hour during operating hours. RPO is 1 hour.

4. IT Security

4.1 API

- API accepts only request over HTTPS
- Solution ensures proper security headers are implemented across all API requests such as Content-type, X-frame-options etc to discourage use of content into third partyframes
- Authentication is either OAuth with AWS Cognito or Personal Access Key/Secret
- Authentication is required for every endpoint, except a few selected public endpoints. Documentation: https://api.envivo.link/docs/latest
- All user input is validated, including data-type checks. User input is sanitized before it can be written to storage (DB/files).

4.2 Frontend / Client Portal

- Pages accept only request over HTTPS
- Solution ensures proper security headers are implemented across all requests such as Content-type, X-frame-options etc to discourage use of content into third partyframes
- Authentication is done against AWS Cognito with Email and Password
- Authentication is required for Portal and all Content Pages
- Authentication is also required to view any uploaded assets like images, pdf documents, videos, etc. These generated assets are stored in private S3 buckets and delivered by pre-signed URLs.
- Users are not able to generate any content by themselves (upload or store any kind of data)
- No visitor data is tracked.

4.3 Backoffice

- Pages accept only request over HTTPS
- Solution ensures proper security headers are implemented across all requests such as Content-type, X-frame-options etc to discourage use of content into third partyframes
- Backoffice App is a JavaScript Client App using the API
- Backoffice is delivered by static webserver
- As all API endpoint require authentication, so does Backoffice App

4.4 Static Assets

- Static Assets like CSS, images and JavaScript Libraries are stored in private S3 Buckets
- Assets are delivered by CloudFront via Origin-Access-Identity over HTTPS
- Assets can only be uploaded or changed by privileged DevOps team members

4.5 User and Client Data

- All personal data is stored either in RDS database (first name, last name, gender, email, mobile phone) or in private S3 buckets (images, etc). All data is encrypted at rest (encrypted RDS instances, encrypted RDS snapshots, and encrypted S3 buckets).
- No customer data (either in RDS or on S3) is publicly accessible without authentication.
- Unauthorized customer data access is prohibited by access control groups and permissions.

4.6 Solution Platform

- Solution does not store any passwords. Authentication is managed by AWS Cognito.
- For Account creation (initial password), email is verified by a registration link (valid for 2 hours). Password is set directly in AWS Cognito. Registration Email is sent by Sendinblue; required data (email, link, first name, last name) is transferred to Sendinblue API over HTTPS
- Password resets are triggered directly in AWS Cognito with PIN code sent by AWS Cognito over AWS SES (Ireland).
- Solution uses AWS platform, thus all resources have appropriate AWS security policy associated with them
- None of the AWS resources are exposed via open API endpoints.

4.7 Infrastructure

- All AWS resources are in VPC and protected by security groups created within AWS
- All ports in AWS resources like EC2 are closed
- AWS console access requires virtual MFA authentication by DevOps team
- Data encryption of AWS snapshots and backups
- EC2 server private keys reset is done regularly
- Only a selected group of DevOps team members has access to critical AWS resources (production environment, CI-CD environment)

4.8 CI – CD

- For Continuous Integration and Continuous Delivery, the GitLab platform is used.

- GitLab is used for code (git) repositories (API, Backend, Backoffice, static Assets), Docker repository, automated tests and automated deployments
- GitLab instance is on secured EC2 instance, backed by encrypted EBS Volume. EBS Volume is backed up twice daily with 7 days retention. Accessible only over HTTPS and ssh.
- Access to GitLab EC2 instance is restricted to selected DevOps team members
- Access to GitLab Resources (git and docker repositories) are private; strong authentication is required; appropriate access levels are granted to development team members based on function and responsibility
- Automated tests are run after each commit
- Automated deployments for dev and staging environments are triggered after successful, manual merges (Code Review by senior Developers)
- A new Deployment builds new Docker images from source, tests these images, and triggers a new ECS Deployment Task. New Docker containers are started on ECS Cluster and replace running containers.
- ECS can access private Docker repository by access tokens, which are rotated regularly
- A deploy to production can only be started after successful deploy to staging, and must be manually confirmed and signed off by selected senior DevOps team members
- Code changes can not be put in production in any other way.