# 2.0 Infrastructure

**Contents of the current section:**



## Provisioning and deployment approach (Terraform)

Terraform is an IAC (**I**nfrastructure **A**s a **C**ode) tool used primarily by DevOps teams to automate various infrastructure tasks. Instead of manually configuring cloud nodes or physical hardware, IaC automates the process infrastructure management through source code. Even if your servers come from different providers, such as AWS or Azure, Terraform helps you build and manage these resources in parallel across providers.

In this project, terraform will be our primary tool to deploy our infrastructure. The code is located in GitHub

Terraform code will create:

- VPC
- EKS cluster (EC2)
- DNS(route53), SSL certificate
- Datastorages (RDS, DynamoDB, S3, ECR)
- Serverless computing (lambda, sqs)
- Operational helm charts (Nginx ingress, Datadog, ArgoCD, ArgoCD-Apps, Cert-manager, Github-actions-controller, ACK-Lambda-controller)
- Bastion host

## Structure and nesting

Code in the Terraform language is stored in plain text files with the .tf file extension. The files located in the root folder describe the highest level of our code structure. These resources presented as a numerical sequence in the .tf  file names. This conditional sequence visually shows deployment levels of the infrastructure:

**01-main.tf** Creates required provides and get prerquesites(terraform version or require provides)

**02-vpc.tf** Creates VPC

**03-key-pair.tf** Creates key pair

**04-eks.tf** Creates EKS cluster

**05-rds.tf** Creates RDS cluster

**06-dynamodb.tf** Creates DynamoDB table

**07-ecr.tf** Creates Elastic Container Registry

**08-s3.tf** Creates S3 buckets

**09-lambda.tf** Creates lambda function

**10-sqs.tf** Creates SQS queue

**11-datadog.tf** Create AWS-datadog integration

**12-helm.tf** Creates operational helm chart resources in the cluster

**13-acm_and_route53.tf** Creates certificate and route 53 records

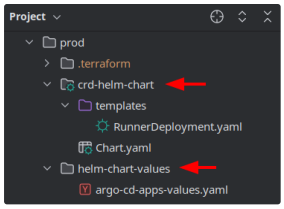
**14-Kubernetes.tf** Creates operational kubernetes resources in the cluster

**15-bastion_host.tf** Creates bastion host for accessing internal resources

Described above file sequence is the highest level of our infrastructure code. If you look inside of them you can find that public modules are used in most of those files.

Inside of these modules you can find a detailed description of our infrastructure. We will look at all of those modules next in this paragraph.

Also, when you look at the file structure you can find files two folders:




**crd-helm-chart forlder**

This is a custom Helm chart that is an integral part of the GitHub Actions controller. This Helm chart will be used to register a self-hosted runner in our CI repository. This process will be described in more detail later in the SOP document.

**helm-chart-values**

This is a custom values YAML storage that will be used by the ArgoCD Apps Helm chart to register the application in the ArgoCD system. This process will be described in more detail later in the SOP document.

Variables:

**variables.tf** This is the one of the most important files. Before started to run code make sure you fill all variables correctly

## Infrastructure building blocks (Terraform modules)

In this section we are going to cover key points in our infrastructure

### VPC

Terraform code will create the following VPC and required route53 DNS records. **2 public and 4 private subnets** are created in 2 different AZs. This enables EKS to protect the workload from any AZ failures. **VPC** is created with IP address 10.0.0.0/16.

```
public_subnets     = ["10.0.4.0/24", "10.0.5.0/24"]
private_subnets    = ["10.0.1.0/24", "10.0.2.0/24"]
database_subnets   = ["10.0.41.0/24", "10.0.42.0/24"]
```

The picture below shows how VPC will be created for our entire project

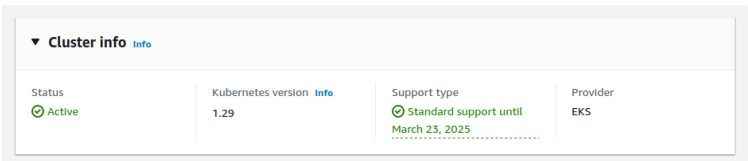

**Internet Gateway** is attached to VPC.

**NAT Gateway with Elastic IP (static IP)** is also placed in the Public Subnet.

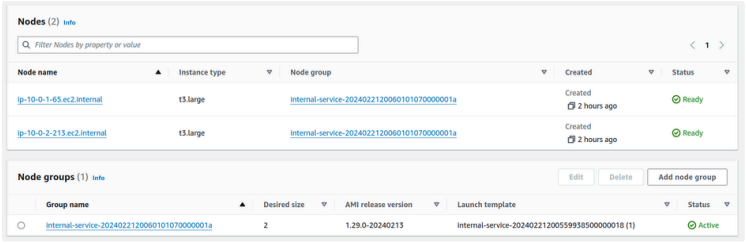
A set of **public, private,** and **main route tables** are also created.

### Orchestration (EKS)

In this section will be going to create our main core it is EKS cluster(EC2). The cluster version is 1.29. All of our applications will be hosted in EKS



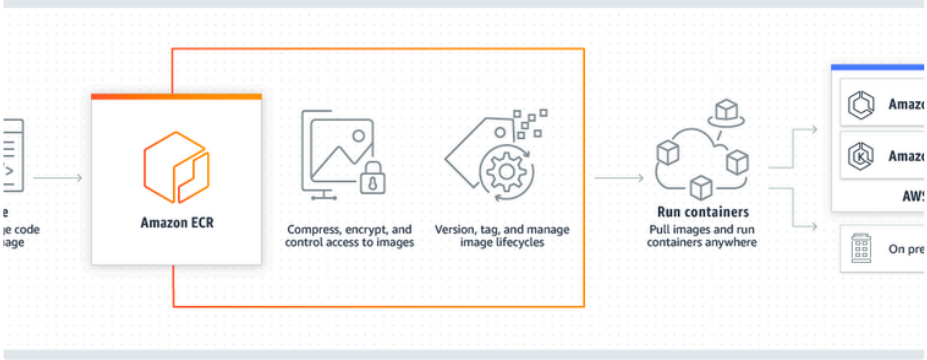

The Kubernetes Cluster using Autoscaler. The cluster autoscaler uses AWS scaling groups. The Kubernetes Cluster Autoscaler automatically adjusts the number of nodes in your cluster when pods fail or are rescheduled onto other nodes. **By default, desired worker node count is 2**



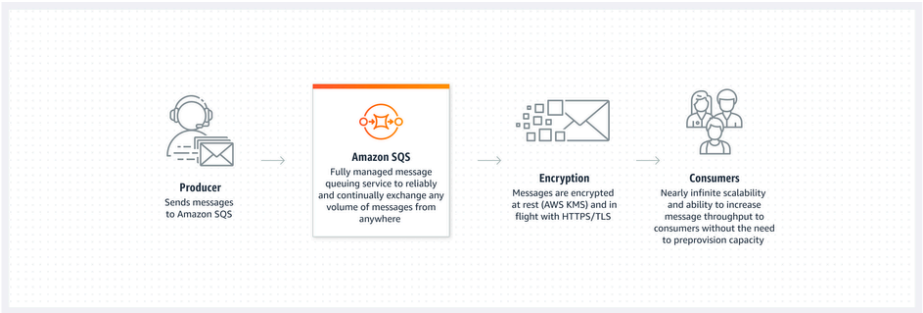

### Datastorages

- **S3** - is an AWS service that provides object storage.
  This storage is used to store PDF files - the results of the application's work.

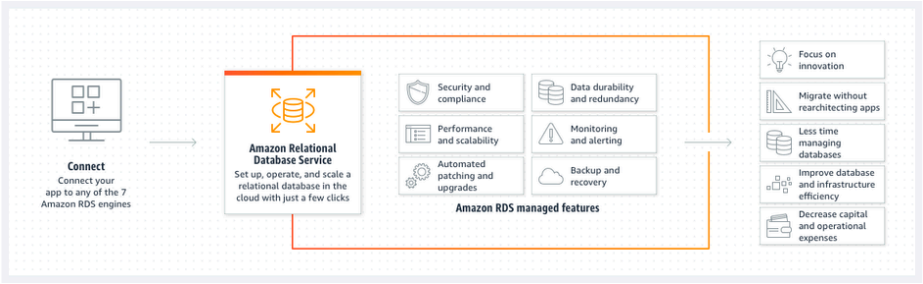

- **ECR** – is an AWS-managed container image registry
  This registry is used to store ready Docker images for API, UI, and Lambda.



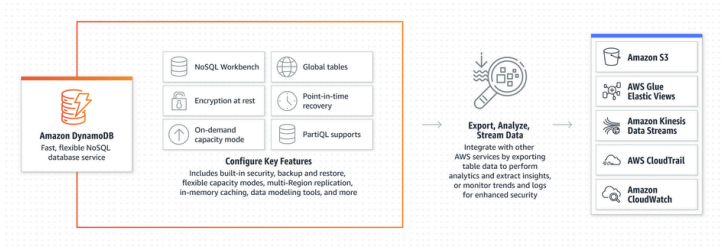

- **SQS - Amazon Simple Queue Service (SQS)** lets you send, store, and receive messages between software components at any volume, without losing messages or requiring other services to be available. It is used to build a queue of requests in the form of messages, while keeping them until successfully processed by the Lambda function.




- **RDS Aurora(Mysql)** - is a collection of managed services that makes it simple to set up, operate, and scale databases in the cloud.
  It is used to implement user authentication data. Our application uses an ORM (GORM), so this type of database is a requirement because it is an interface for interacting with relational databases.




- **DynamoDB** is a serverless, NoSQL, fully managed database service with single-digit millisecond response times at any scale, enabling you to develop and run modern applications while only paying for what you use. DynamoDB is used for two reasons. The first is to diversify our project for educational purposes, and the second is to speed up the work of both Lambda functions and the application as a whole. It provides low latency, high read/write rates, and auto-scaling when working with data. User data is stored in this database in the form of a catalog of processed files for a specific user.




### Operational helm charts (Nginx ingress, Datadog, ArgoCD, ArgoCD-Apps, Cert-manager, Github-actions-controller, ACK-Lambda-controller)

The next step will be creating some vital entities in the EKS cluster using terraform via helm. This list of services will be:

- **Nginx Ingress Controller** - is an Ingress controller for Kubernetes using NGINX as a reverse proxy and load balancer
- **Datadog -** Datadog agent for monitoring kubernetes cluster
- **Kubernetes Cluster Autoscaler** - It ensures that your cluster has enough nodes to schedule your pods without wasting resources.
- **ArgoCD -** responsible for continuously monitoring all running applications and comparing their live state to the desired state specified in the Git repository
- **ArgoCD-Apps -** responsible for managing Argo CD Applications and Projects.
- **Github-actions-controller -** Kubernetes operator that orchestrates and scales self-hosted runners for GitHub Actions
- **ACK-Lambda-controller -** allows you to manage AWS services directly from Kubernetes (lambda, sqs etc)
- **Cert-manager -** is a powerful and extensible X.509 certificate controller for Kubernetes in our project it is dependency for github-actions-controller.

**The most important of them are Nginx Ingress Controller.**

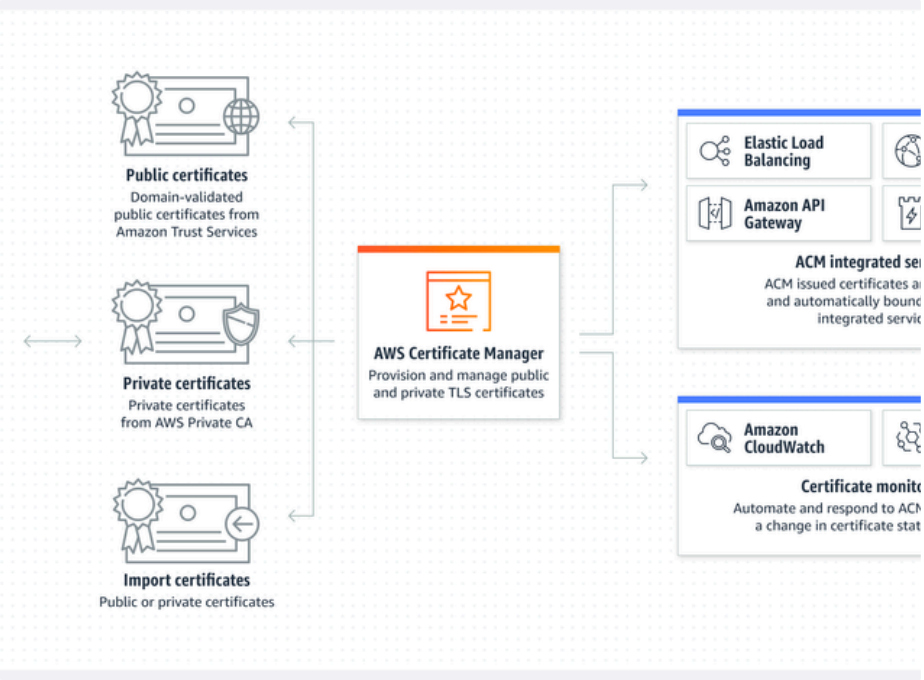
**Nginx ingress controller -** It is a Kubernetes controller that manages the Ingress resources and acts as a reverse proxy to route incoming HTTP and HTTPS traffic to the appropriate services within a cluster. It enables external access to applications running in the cluster.

**Datadog** - is an observability service for cloud-scale applications, providing monitoring of servers, databases, tools, and services, through a SaaS-based data analytics platformLast but not least is the **Bastion host -** A bastion host is a special-purpose computer on a network specifically designed and configured to withstand attacks

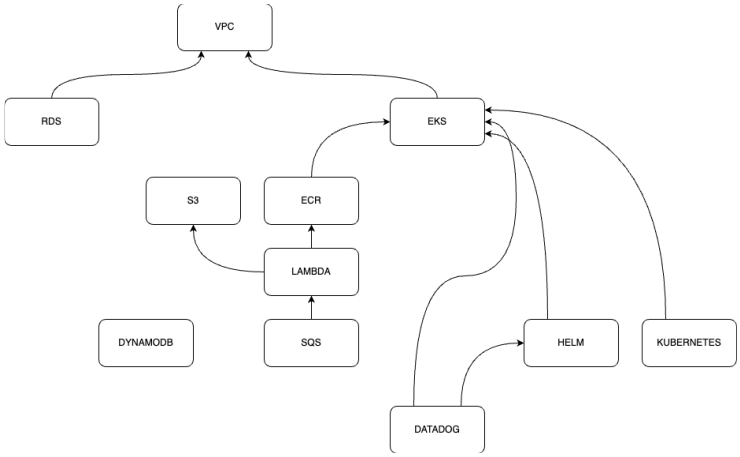
### DNS(route53), ACM(amazon certificate manager)

For managine DNS records we are using route53. Amazon Route 53 is a highly available and scalable Domain Name System (DNS) web service.

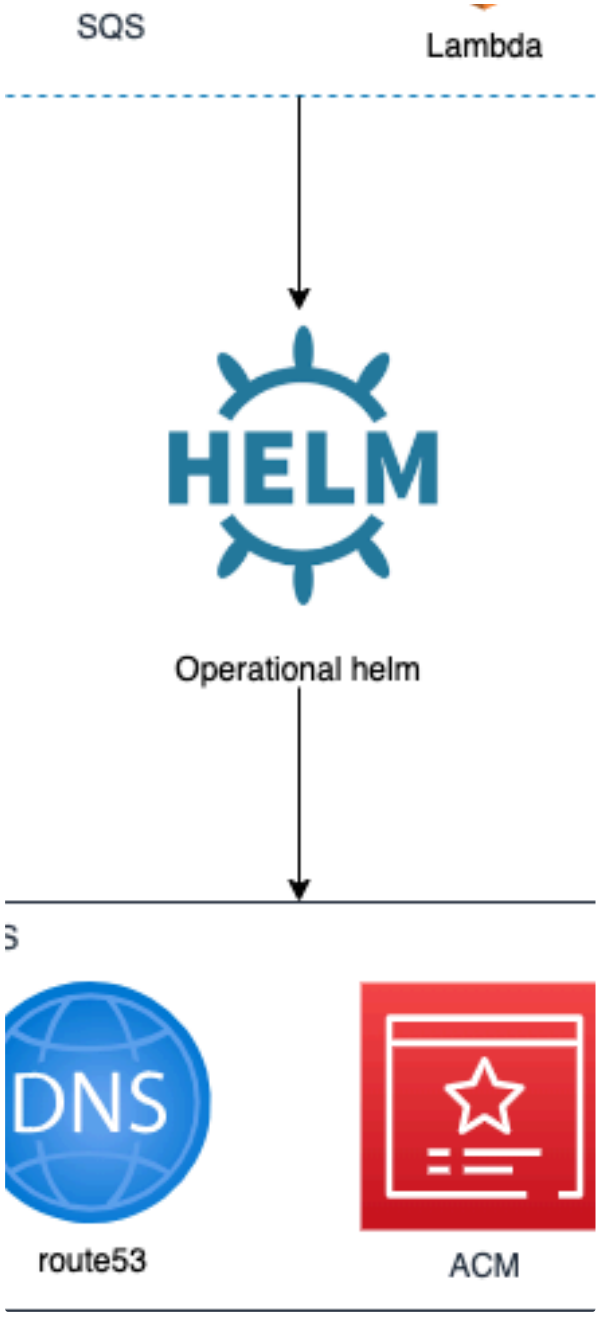
For manging ssl/tls ceritifcate best option is to use AWS ACM. AWS Certificate Manager (ACM) to provision, manage, and deploy public and private SSL/TLS certificates for use with AWS services and your internal connected resources




Terraform dependency graph

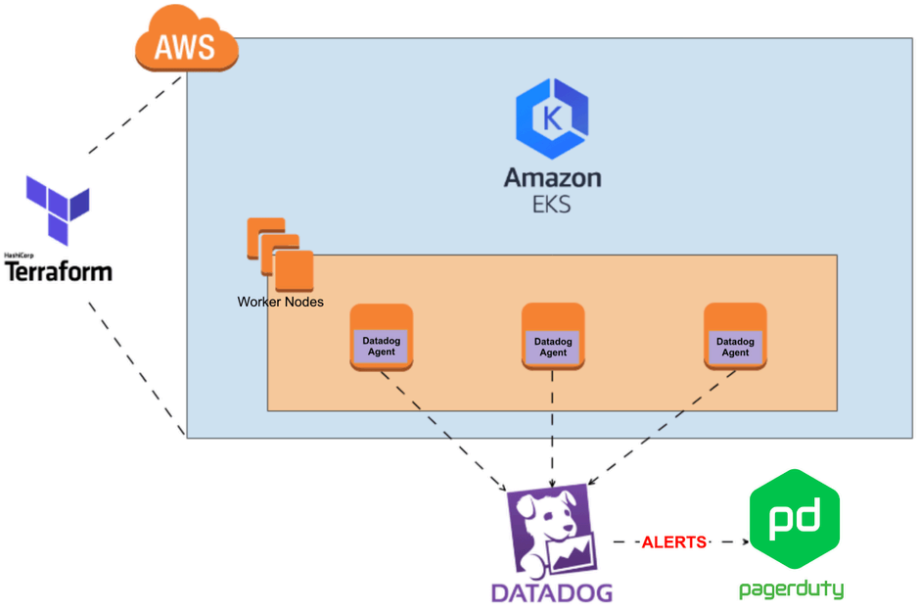


Modules dependency graph




## Monitoring and logging (Datadog)

**Datadog -** Datadog is a monitoring and analytics tool for information technology (IT) and DevOps teams that can be used to determine performance metrics as well as event monitoring for infrastructure and cloud services.




## Incident management (PagerDuty)

**Pagerduty** is an operations performance platform that provides IT alert monitoring, on-call scheduling, escalation policies used to fix problems in apps, servers and websites across the entire incident lifecycle