TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Song Huong Pham Thi IVSB184073

Converting Legacy Application to the Cloud: The case of Certidude

Bachelor thesis

Supervisor: Lauri Võsandi
MSc
Toomas Lepikult
PhD

Tallinn 2021

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Song Huong Pham Thi IVSB184073

Pärandrakenduse viimine pilvekeskkonda Certidude'i näitel

Bakalaureusetöö

Juhendaja: Lauri Võsandi
MSc
Toomas Lepikult
PhD

Tallinn 2021

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Song Huong Pham Thi

29.04.2021

Abstract

The goal of this thesis is to design cloud architecture that is secure, highly available, scalable, and performant for Certidude – a certificate authority management application.

Certidude was originally developed as a monolith application. Although now it has been through some architecture changes to facilitate replication, it’s still not clear how and if it is possible to deploy Certidude on the cloud.

The thesis analyzes public articles of similar application cloud solution to identify the best option, documents the implementation process and analyzes the result.

This thesis is written in English and is 47 pages long, including 5 chapters, 12 figures, and 3 tables.

Annotatsioon

Lõputöö eesmärk on disainida pilvearhitektuur mis on nii turvaline, kõrge kättesaadavusega, skaleeruv kui ka tõhus, et seda saaks kasutada Certitude – sertifikaadi väljastamise tarkvara.

Certidude loodi algselt monoliitrakendusena. Kuigi see on nüüdseks on see läbinud mõned arhitektuurilised muutused eesmärgiga hõlbustada replikatsiooni, hetkel pole veel kindel kas ja kuidas oleks võimalik Certidude'i pilvekeskkonda paigaldada.

Lõputöö analüüsib avalikke artikleid, mis käsitlevad sarnaseid pilverakenduste lahendusi, tuvastamaks parima valiku, dokumenteerides töö teostuse protsessi ja analüüsides tulemusi.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 47 leheküljel, 5 peatükki, 12 joonist ja 3 tabelit.

List of abbreviations and terms

AWS	Amazon Web Services
ECR	Elastic Container Registry
ECS	Elastic Container Service
SLA	Service Legal Agreement
CRL	Certificate Revocation List
LDAP	Lightweight Directory Access Protocol
ALB	Application Load Balancer
CSR	Certificate Signing Request
ELB	Elastic Load Balancer
NLB	Network Load Balancer
PKI	Public Key Infrastructure
CA	Certificate Authority
HSM	Hardware Security Module
KMS	Key Management Service
IKEv2	Internet Key Exchange version 2
EC	Electronic Certificate
RA	Registration Authority
VA	Validation Authority

Table of Contents

List of Figures

Index of Tables

Introduction

Cloud computing is the delivery of computing resources on demand, taking away users' responsibilities to administrate, maintain and secure underlying hardware. Nowadays, many businesses are moving towards cloud computing for scaling and resource allocations. This has opened a lot of oppoturnities to new businesses because this cut their management, hardware, maintanance costs greatly. Furthermore, developers could focus on developing applications by delegating the responsibilities of server management to cloud providers.

Certidude is a certificate authority management application that was originally developed as a monolith application. Since many users of Certidude want to take advantage of the cloud, this paper will analyze the possible options and offer a cloud solution for Certidude with a focus on security.

1 Background

Deploying applications on the cloud is nowadays more relevant than ever and applications that have been developed in traditional setting have issues that are most obvious when attempting to deploy them on cloud. This chapter will give an overview of what is the cloud, AWS as the chosen cloud provider, Certidude, and answer why it wants to be on the cloud.

1.1 Cloud computing

1.1.1 Why the cloud

Nowadays, many businesses chooses the cloud as the infrastructure, because the benefit of no upfront investment in hardware and servers which help them start small and grow with their need.

1.1.2 Amazon Web Services

Amazon is the first to offer cloud computing services. Currently, AWS is one of the largest cloud providers, offering a variety of services such as compute power, database, content delivery network, cache, and many more.

Since it is decoupled into docker containers, deploying Certidude would involve mainly AWS container services such as ECR, ECS, Fargate, database service such as DynamoDB or self-hosted mongoDB, application load balancer, CloudFront for static content.

ECS is a container orchestration service. An open-source popular orchestration service to ECS is Kubernetes.
ECR is a managed container registry that is similar to Docker Hub.

Fargate is a a serverless compute engine for containers. Instead of running containers on EC2 instance, running on Fargate helps reduce provision and management effort.

1.2 Certidude

Certidude is a Certificate Authority management tool for OpenVPN operators, developed by Koodur OÜ to ease the process of setting up VPN clients, managing and revoking VPN certificates.

Certidude was developed to be VPN technology neutral, currently OpenVPN and StrongSwan (IKEv2) are supported, but it's trivial to add, for example, WireGuard support as the general framework is very similar.

Some of Certidude's main features include certificate request submitting, certificate signing and access revoking integrating with StrongSwan/OpenVPN gateway, NetworkManager for Fedora and Ubuntu.

1.2.1 Why Certidude on the cloud

Koodur OÜ has been approached by several companies about connecting IoT devices to their AWS infrastructure, but without easy options for deploying Certidude on cloud that never materialized in any real business. Moreover, the benefit of deploying application to the cloud, scalability, availability is pretty much obvious as those are handled by service provider. Other problems for on-premise deployment such as installing software, hardware, backup, disaster recovery, etc is also removed by using cloud. In addition, with a cloud deployment you get the ability to easily reach almost all parts of the world.

1.2.2 Functionalities, use cases and architecture

A roadwarrior wants to access services behind OpenVPN/ StrongSwan gateway so he submits his CSR to Certidude in order for it to be signed. An administrator sees the CSR, examines it and signs it. When a roadwarrior submits its certificate to OpenVPN/ StrongSwan gateway, it asks Certidude and Certidude asks Domain Controller for the

user whitelist, then answers the gateway. The gateway then allows or declines the access request of the client.

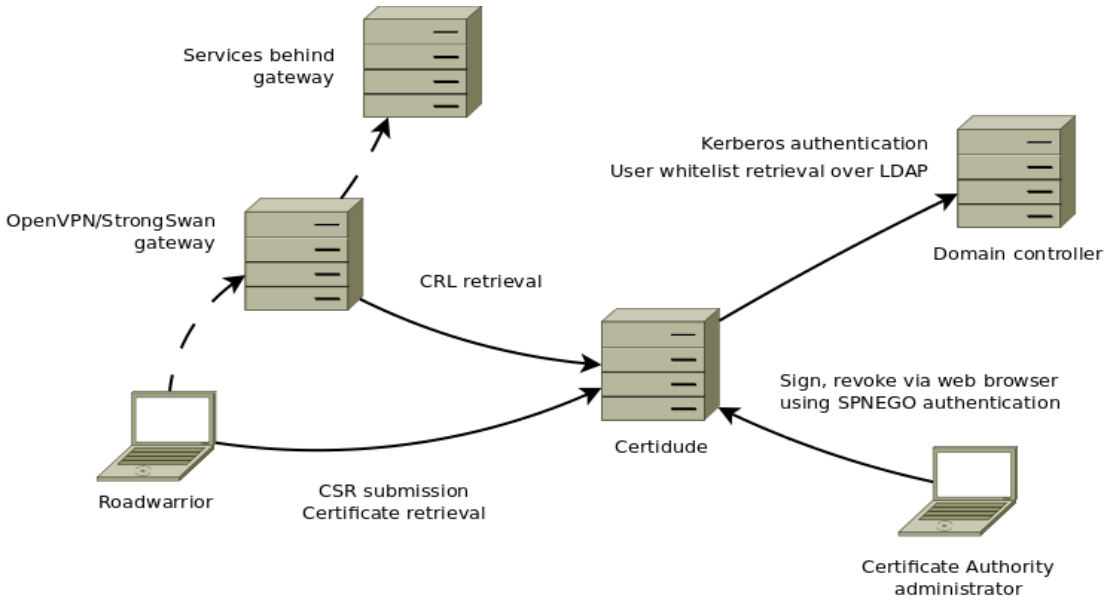


Figure 1: Certidude overview [27]

Some current main use cases of Certidude:

- Provide a simplified on-prem self-hosted VPN enrolment system.
- On-prem self-hosted VPN gateway for IoT devices by using GL iNet AR150 routers as IPSec VPN clients that proxy traffic to network enabled printers.

Overview of Certidude architecture:

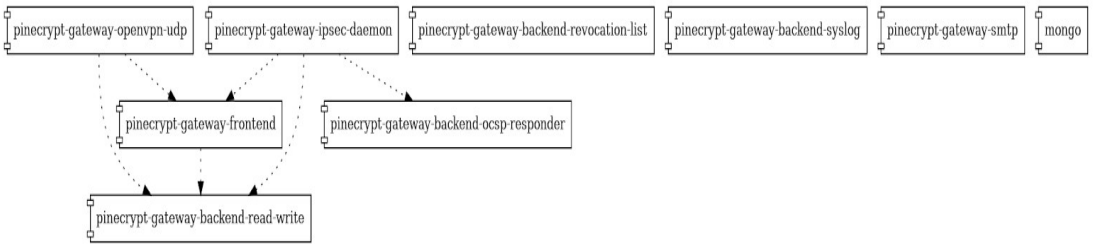


Figure 2: Certidude containers (Source: Author created)

2 Description of problem and development requirements

This chapter will describe the problem that previously prevented Certidude ability to be deployed on the cloud, what has been further developed since then, the current problem, the requirements from the product owner side and the goal of this thesis.

2.1 Description of problem

Previously Certidude was not dockerized, so it could only be installed manually by apt-get and pip install which was tedious and error prone. Moreover, the database was a directory in local filesystem. However, for organizations with many clients, such setup is not scalable enough to handle all the traffic load. The cloud is specifically designed to tackle this problem because of its high scalability. Furthermore, it also offers high availability, resilience, security as well as performance. With the cloud, users don't have to worry about server maintenance or potential server failure because these responsibilities have been taken by cloud providers.

Now that Certidude has gone through development changes with Docker and MongoDB, the deployment is easier and replicating data between nodes is easy. However, it's still not clear if what needs to be modified further for deployment in AWS. Moreover, deploying PKI on the cloud carries the risk of critical security infrastructure to the cloud.

The goal of current thesis is:

- Analyze the possible options of AWS services that could be used to deploy Certidude on the cloud.
- Document the implementation process and implementation issues.
- Recommended option and point out the pros and cons of these based on reliability, security, performance and cost optimization.

- Further investigate possible AWS services that could enhance the security of Certidude on the cloud.

2.2 Development requirements

Some requirements for Certidude architecture specified by Koodur OÜ are:

- The chosen cloud provider is AWS.
- Database is MongoDB.
- It should be possible to run more than 2 replicas of Certidude stack in different geographical locations for high availability.
- Final result must support connecting from Windows workstations using IPSec IKEv2 using EC certificates.
- the refactored software must remain deployable on traditional infrastructure (for example Vmware).

3 Review and analysis

3.1 Review of public articles

3.1.1 Monolith to microservices

Here is Telia Customer Information Management Platform Cloud Solution [3]. Telia CIM is also previously a monolith application, then moved on to the cloud and also using container service like Certidude.

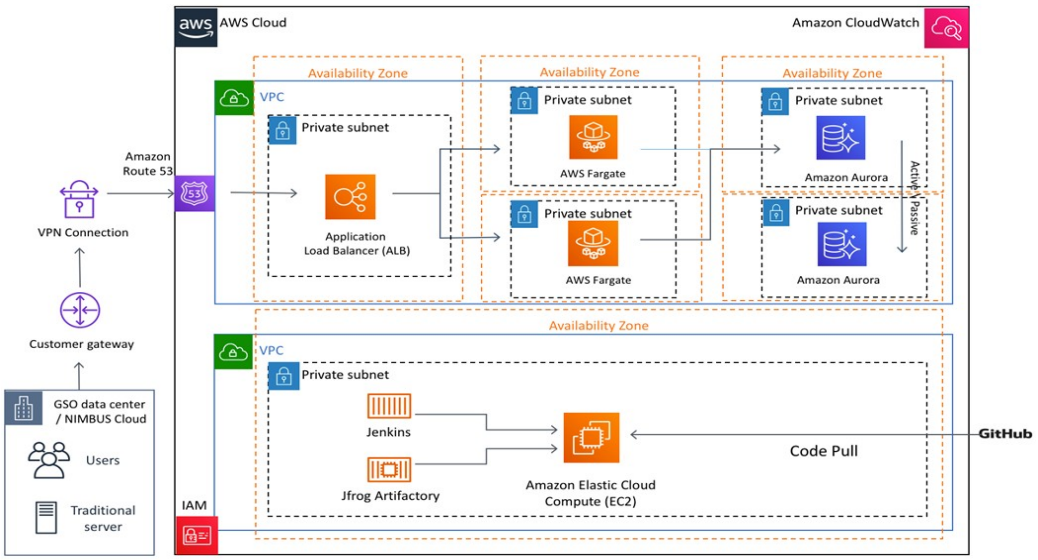


Figure 3: Telia CIM architecture [3]

From this architecture, how container services are separated in different subnets and availability zones, how containers connect to database and also talk to users.

Typical microservices application on AWS :

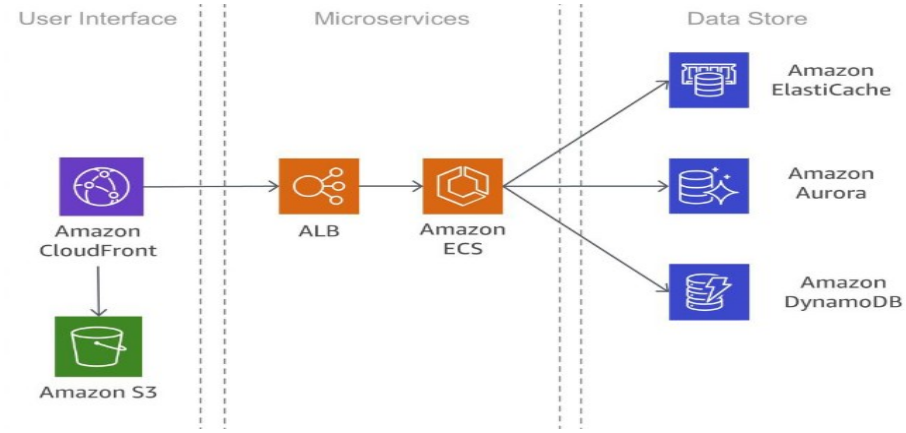


Figure 4: example of typical AWS microservices [22]

From this example, static content for user interface is served with CloudFront and S3. ECS acts as entry point for applications logic behind set of programming interface and data is persisted through DynamoDB.

Further challenges of microservice architecture are:

- Service discovery.
- Monitoring.
- Caching.
- Auditing.
- Notification and queueing.
- Failover between instances.

Service discovery is how services communicate and interact with each other. Not only microserivce distributed characteristics make communication harder but also impose other problems such as health check, how and when to store configuration data [22]. Some techniques to solve this problem could be DNS-Based service discovery or third-party software such as HashiCorp Consul, etcd, or Netflix Eureka [22].

For a distributed system, many parts need to be monitored. AWS CloudWatch could be used to collect and centralise logs. Especially for Fargate where direct access to server is not available. There is also another popular option which is Prometheus.

Caching is a way to reduce latencies in application. Many applications use ElasticCache to reduce the volume calls to microservices by caching locally [22].

Auditing helps enforce security policies because it gives an overview of user actions on each service and a good overview of all services at organizational level [22].

From these examples, some challenges with moving from monolith to microservices architecture on the cloud are fore-seen. Yet there's still a need to analyse more similar applications to Certidude, specifically CA cloud solutions for a clearer approach of how to deploy Certidude.

3.1.2 PrimeKey EJBCA

PrimeKey EJBCA is the world's most used certificate issuing and management software [30]. Along with its service, PrimeKey EJBCA also cloud solution that is available on AWS marketplace. Here is its reference architecture:

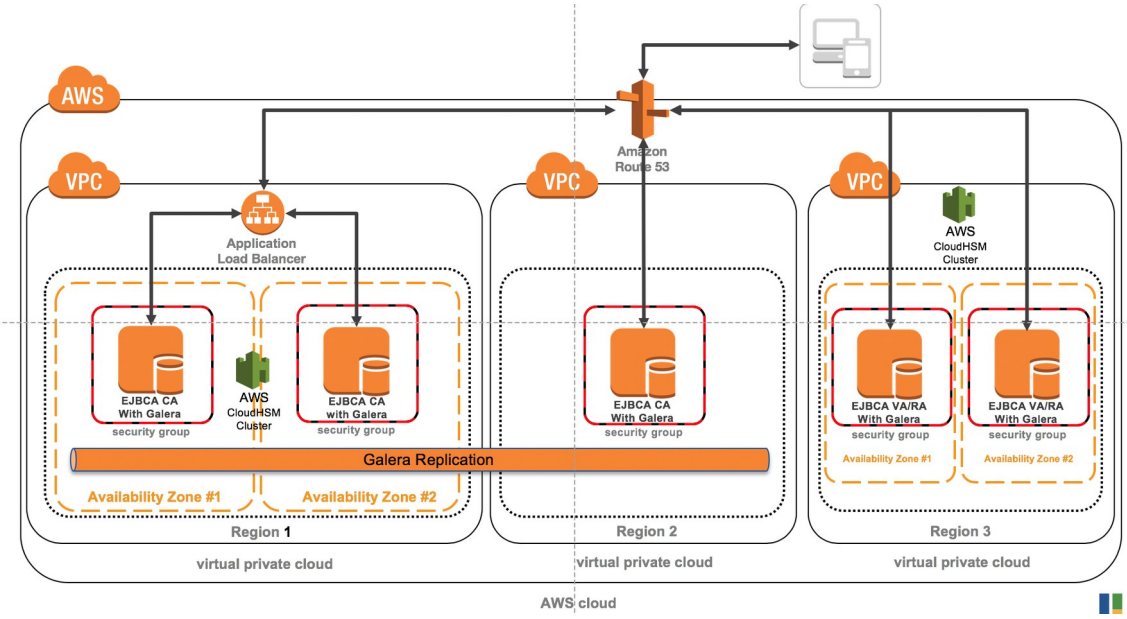


Figure 5: PKI on AWS – PrimeKey EJBCA [32]

The architecture includes [32] :

Site level redundancy leveraging availability zones in region 1.

- Galera replication configured manually across all regions for active/active CAs.
- Application Load balancer (ELB) for redundancy in region 1.
- Amazon Route 53 load balancing across all remaining sites and to ELB.
- Security groups protecting all nodes at each site.
- VA/RA services in separate availability zones.

What are learned from this architecture are:

- ELB should be used for region replication.
- CloudHSM for backing keys.
- Route53 for load balancing.
- Each replica is in different security group.

3.1.3 Microsoft PKI Infrastructure on AWS

Microsoft has its own PKI that is deployed on AWS. Although this is windows-hosted PKI infrastructure, it's still a helpful reference as many AWS services in this architecture are used so given customers more options to analyse and investigate.

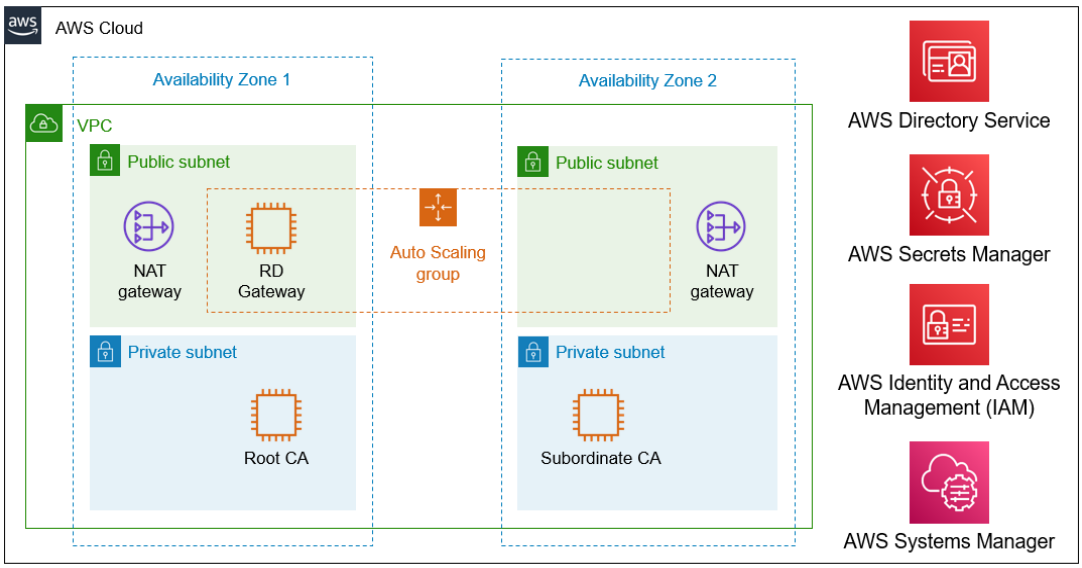


Figure 6: Microsoft PKI on AWS [21]

What is learned from this article are:

- VPC should be configured with private and public subnets for security.
- auto-scaling group for high availability.
- Secrets Manager to store credentials.
- System Manager could be used to automate CA deployment process and store generated certificates.
- IAM to enable EC2 instances and System Manager.

3.1.4 Empathy OpenVPN CA AWS Architecture

Empathy is a company that is using open-source OpenVPN CA on AWS. Here is how the CA is deployed:

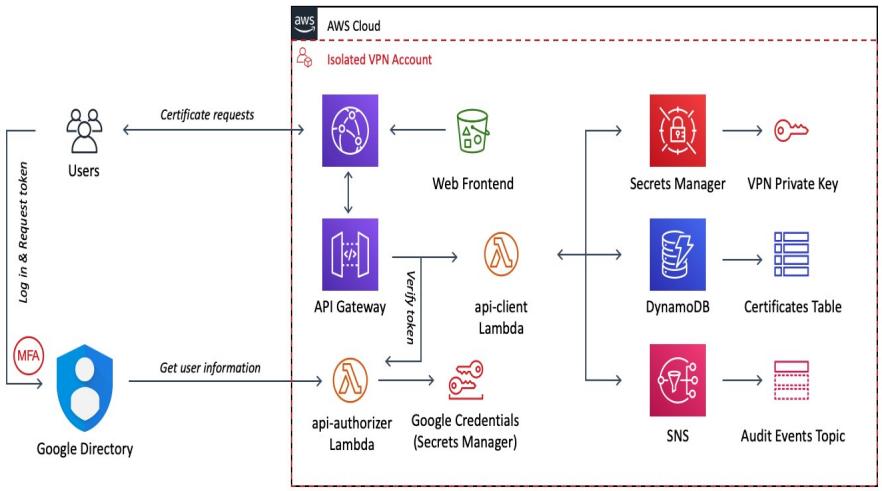


Figure 7: Empathy OpenVPN CA architecture [29]

This architecture includes:

- Secrets Manager containing the Google Service Account Key used for querying the Directory (this is the Google Cloud equivalent to an Access Key ID + Secret Key pair for an AWS IAM User). [30]

- DynamoDB table that stores all certificates that have been signed by the CA and are still valid. [30]
- Lambdas for the Client and Server APIs, Authentication, key rotation, event processing, etc. [30]
- API Gateway to tie all the Lambdas together on a public API. [30]
- S3 Bucket with the Web Frontend release code. [30]
- CloudFront distribution with both the S3 Bucket and API Gateway as Origins. It acts as the main entry point to the system. [30]
- SNS topic that receives all audit events of the system. It has Lambdas subscribed that process all the events, log them, and alert on suspicious activity. [30]

Here Google Directory presents the organisations' user directory. Certidude also expects that organisations have their own user directory.

From this example, what could be applied to Certidude are:

- Secrets Manager for Privacy Key rotation.
- Route53 for hosted zone.
- Lambda could be used for server and client API.
- S3 + CloudFront for frontend.
- API Gateway for interacting with users.
- SNS topic for event pushing.

However, these are all generic CA architectures so they don't address to cloud infrastructure simplified setup for connecting IoT devices, while Certidude is like less of managing CA but more about getting VPN connections up and running from variety of platforms such as Windows, IOS, Android, Linux, OpenWrt, etc.

3.2 Analysis of different options

This section will analyse different options for container service, databases, TLS termination, event-pushing to browsers, serving static assets, service discovery, options for storing keys, and security.

3.2.1 EC2 vs ECS Fargate vs ECS EC2

As Certidude is developed with containers, a cloud container service option is needed to use for deploying Certidude. In AWS, there are 3 options available EC2, ECS EC2 and ECS Fargate. Here is a comparison table between ECS Fargate and ECS EC2:

Table 1: ECS Fargate vs ECS EC2 [26]

	ECS Fargate	**ECS EC2**
Host OS	Linux, Windows	Linux
Max vCPU	4	448
Max Memory	30 GB	26 TB
CPU bursting	no	yes
Pricing	Per running EC2 instance	Per running task
Discount	Compute Savings Plans, Spot Instance	Reserved Instances, Savings Plans, Spot Instance
Operation effort	Low	High
Networking options	Multiple	ENI per task

Hosting containers on EC2 is pretty much the same with hosting containers on local machine. The difference between ECS EC2 and EC2 is that ECS EC2 allows launching a cluster of machines that will serve as the deployment ground of container apps, allowing to treat all instances in the cluster as one big instance available for container workload, while with EC2 you have to manage each instance separately and maintain the connections between servers yourself.

Although using serverless technology like Fargate could be very useful as we don't have to guess the CPU, memory limit of each containers, its task definition doesn't support *privileged* and *devices* [21]. Certidude is for managing VPN certificates, we can't assign IP address */dev/net/tun* and *NET_ADMIN* to it as configuration so Fargate is not a viable option in this case.

3.2.2 DynamoDB vs self-hosted Mongo vs DocumentDB

Comparison between DocumentDB and Mongo:

Table 2: self-hosted MongoDB vs DynamoDB [28]

	self-hosted MongoDB	**DynamoDB**
Primary database model	Document store	Document store Key-value store
Configuration	All are permitted	Few are not permitted
Access to underlying machine	Yes	No
Backup	Self-responsible	Automated
Basic maintenance	Self-responsible	Automated
Same setup for development, test and production	Yes	No
Extra work	Extra-work to persist data across containers	No
Share machine for workload	If workload is small, a machine could be shared for multiple processes	No
Scaling	Equal to the time a new EC2	Instantly

	instance start-up	
Pricing	Cheap	Depends on document retrieval

DynamoDB is an integrated AWS services so it's easier to develop an end-to-end solution. It goes without saying that DynamoDB will offer many advantages coming along with cloud provider such as full-managed server, out-of-the-box security, well-integrated with other services. However, DynamoDB is vendor-specific – using it might dismiss the multi-cloud strategy. Moreover, Certidude has grown large to depend on pymongo. Using DynamoDB meaning that lots of code changes need to be applied to move from *pymongo* to *dynamongo* which is the python package to interact with DynamoDB.

DocumentDB is the middle ground between MongoDB and DynamoDB. The main differences between DocumentDB and MongoDB are that when scaling the time it takes for MongoDB is equal to the time a new EC2 instance starts, while for DocumentDB it's pretty much instantly. Of course, that comes with a cost, price for DocumentDB is very expensive while for self-hosted MongoDB it costs as much as EC2.

In Certidude case, since the project has grown big and depends on pymongo, if organization doesn't mind to pay more DocumentDB is recommended.

3.2.3 TLS termination

TLS termination is mainly about web UI HTTPS portion. This not applicable to OpenVPN/IPSec because OpenVPN uses primarily UDP and not TCP and IPSec uses IKEv2 which uses UDP. AWS could handle this with Elastic Load Balancer. However, currently Certidude is already developed for each containers so every replica runs nginx accepts HTTPS connection, decrypts and forwards it to backend.

3.2.4 Pushing events to browsers

Currently, Certidude actively watches events in Mongo and then proxies that information to browsers and also push signed certs to clients using longpoll. Some AWS options for pushing events are SNS and SQS and MQ but this comes with the cost of vendor lock-in. One difference between these two options is that AWS MQ won't need code changes but more expensive and harder to scale [23]. But so far, no reason has been found why Certidude developers should move from Mongo to any of these AWS services.

3.2.5 Serving static assets

Static contents such as JS and CSS could be served by CloudFront from S3. However, in some cases this approach might actually add latency [23]. A better way might be to implement other caching mechanisms to reduce chattiness and minize latency [23].

3.2.6 App Mesh vs NLB vs ALB

In order to further monitor and control microservices, AWS App Mesh is used. App Mesh is a service mesh that provides application-level networking for services communication between different type of compute infrastructure [16]. One similar popular service mesh tool is Istio.

However, load balancer is a good alternative. With the complexity of App Mesh, few advantages that it offers aren't worth the cost. Here is a comparison table between difference service discovery options AppMesh, ALB and NLB:

Table 3: App Mesh vs ALB vs NLB [25]

	App Mesh	**ALB**	**NLB**
Observability	Inbound, outbound requests	inbound requests only	very little insights
Fault Tolerance	retries, circuit Breaker	client-side code needed	client-side code needed
Resource Efficiency	1-3 sidecar containers per task/pod	Yes	Yes

Fully Managed	Only parts of the service	Yes	Yes
Costs	Cost for CPU and memory of sidecar containers	Hourly fee traffic	Hourly fee traffic
Complexity	Additional layer of abstraction	simple	simple

At the time of comparing different options for service discovery, it was still thought that it was good practice for each containers to be in separate task definition for further replication of each services when needed to save CPU and memory. Hence, the analysis of service discovery options. However, later on it's discovered that all containers should be in the same task definitions because of the networking complexity between containers especially for CA software like Certidude so microservice service discovery is now removed.

3.2.7 CloudHSM vs KMS

KMS is Amazon's managed encryption and key management service that creates and stores the cryptographic keys and uses AWS infrastructure for signing operations. Amazon KMS operations are always backed by HSMs. [1]

CloudHSM is a cloud-based hardware security module (HSM) that enables you to easily generate and use your own encryption keys on the AWS Cloud. [1]

The difference between KSM and CloudHSM is that KSM is shared hardware tenancy - keys are in their own partition of an encryption module shared with other AWS customers, each with their own isolated partition, while Cloud HSM gives you your own hardware module [1]. Additionally, AWS KSM only uses symmetric keys, while CloudHSM allows symmetric and asymmetric keys [1].

For Certidude case, since it's better to ensure keys are isolated on their own encryption module for compliance purposes, it's better to use CloudHSM.

3.2.8 Security enhancements

Some suggestions that could be investigated further to enhance Certidude security:

- Each services have their own security groups to make sure that only necessary rules are allowed.
- Isolation of the CA system on a separate AWS account, ensuring no-one may access the Private Keys of the CA or any other part of the system. [30]
- AWS Config could be used to define security policies and detect, track and alert policy violations.
- AWS CloudHSM could be used for keys.
- Periodic rotation of the CA keys (every month) and low validity for client and server Certificates (1 month) following best industry practices. [30]
- Each replica should be in different security group.
- In addition to LDAP, Oauth or OpenID could be implemented for further authentication.

3.3 Analysis of chosen options

At the moment, the architecture for Certidude hasn't been actually implemented yet so it's not possible to analyse the exact services that are chosen for Certidude. Rather, this section will analyse the recommended ones which are:

- ECS EC2 as container service
- DocumentDB for database
- Route53 for hosted zones
- CloudHSM for encrypted keys
- Secrets Manager for protecting access to applications
- API Gateway
- S3 + CloudFront for frontend

- ELB for load-balancing and replication

The analysis includes reliability, security and cost optimization.

3.3.1 Reliability

Reliability is the ability to perform intended function correctly and consistently when it's expected to [12]. The designed principles for reliability are [4]:

- Automatically recover from failure.
- Test recovery procedures.
- Scale horizontally to increase aggregate workload availability.
- Stop guessing capacity.
- Manage change in automation.

For EC2 which applies for ECS EC2 and self-hosted Mongo in Certidude case, Amazon guarantee 90% uptime. Notice that 90% uptime is for a year so this means that in a year around 36 days the service could be down, which is huge for any organization applications. For DocumentDB, the guaranteed uptime is 99.99%.
Another Certidude replica will also be deployed in another region for high-availability.

3.3.2 Security

Design principles for security are [4]:

- Implement a strong identity foundation.
- Enable traceability.
- Apply security at all layers.
- Automate security best practices.
- Protect data in transit and at rest.
- Keep people away from data.
- Prepare for security events.

Security is ensured by providing service-to-service authentication and authorization with encrypted data in transit. CloudHSM is used for managing encryption keys.

3.3.3 Cost optimization

Design principles for cost optimization are [4]:

- Adopt a consumption model.
- Measure overall efficiency.
- Stop spending money on undifferentiated heavy lifting.
- Analyze and attribute expenditure.

For ECS EC2, there is no additional charge for using ECS as the underlying power of ECS is EC2 - the cost depends on EC2 instance type and used resources. However, there is still potentially unused resources with EC2 which we need to pay for. There are several cost optimization options such as spot instances, reserved instances and saving plan.
Spot instances are unused EC2 instance that is available for less than on-demand price [15]. This makes the bill significantly cheaper about 50% or more. However, the spot workload could be terminated at anytime.
Reserved instances are instances that you pay upfront for a specific amount of time which provides a significant discount (up to 72%) compared to On-Demand pricing and provide a capacity reservation when used in a specific Availability Zone [17].

Saving plan is a flexible pricing model offering lower prices compared to On-Demand pricing, in exchange for a specific usage commitment (measured in $/hour) for a one or three-year period [18].

Although AWS gives out free-tier eligible for a year for Micro instances, their performance is not enough for Certidude so this is not an option.

For DocumentDB price is very expensive – a single server can cost around $200 [30].
Hosting your own MongoDB on an EC2 is much cheaper but then again waiting time for instance replication and management effort are much more.
If Certidude developers decide to move on to DynamoDB for helping organization reduce operation cost, the price could be much cheaper.

Empathy open-sourced OpenVPN CA on AWS which is discussed in review public article section, is used here as a reference [30]:

- EC2 Instances (2x t3.nano Spot): $2.44.
- EC2 Instances (2x 8GB EBS): $1.76.
- Secrets Manager (2 secrets): $0.80.
- Secrets Manager (API Calls): $0.00.
- Route53 Hosted Zone: $0.50.
- DynamoDB (On-Demand): $0.00.
- DynamoDB (Storage + Streams): $0.00.
- Lambda executions: $0.00.
- API Gateway: $0.00.
- S3 + CloudFront: $0.00.
- SNS: $0.00.
- Total: $5.50. (per month for 50 users, 8 hours/day, 20 days/month on 2 Availability Zones [30])

Empathy Open-sourced OpenVPN CA is pretty similar to Certidude because in Certidude EC2, Secrets Manager, Route53, S3, CloudFront will also be used. However, some difference are DynamoDB and Lambda are not yet decided to be used or not.

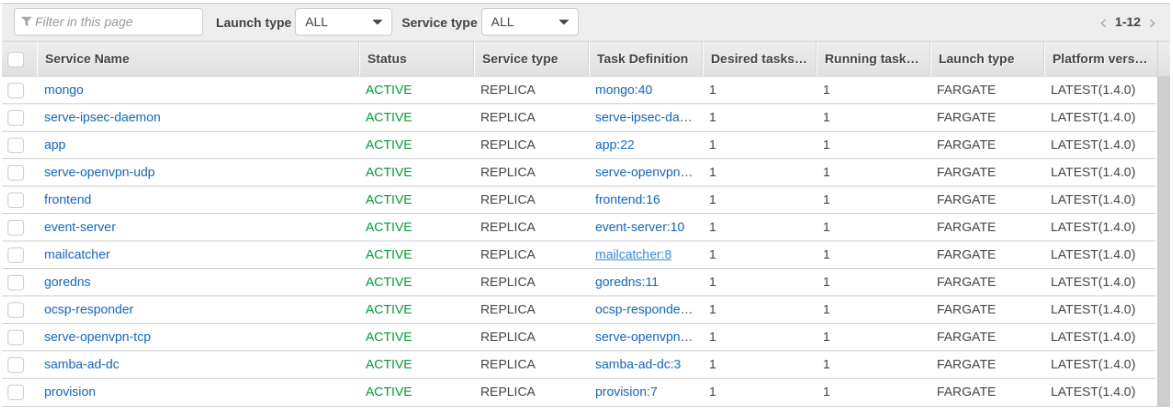
4 Implementation

4.1 Deploying progress

Firstly, IAM users are created. Using an IAM account instead of a root account is one of AWS Security Best Practices in IAM [22] since having a root account access key will give full access to AWS resources and permissions of the root account could not be reduced [22].

To run containers, Certidude cluster was created in container orchestration service ECS. Inside the cluster, there are tasks that AWS containers run within. Tasks are declared by task definitions which are a group of one or more container configurations that define metrics such as CPU, memory, log, environment variables, data volumes, etc [31]. These task definitions were generated by a third-party utility called Container-transform. Container-transform is a small utility to transform various docker container formats to another [6]. Currently, it supports Kubernetes Pod specs, ECS task definitions, Docker-compose configuration files, Marathon Application Definitions or Groups of Applications, and Chronos Task Definitions [32]. Here Container-transform is used to convert docker-compose to ECS task definition.

Service is used to guaranteed that many tasks would be run at all time [31]. If one task under a service fails, another task is automatically deployed until the desired task count is reached. For high availability, each services is scaled up to 4 tasks.



Filter in this page | Launch type: ALL | Service type: ALL | 1-12

Service Name	Status	Service type	Task Definition	Desired tasks...	Running task...	Launch type	Platform vers...
mongo	ACTIVE	REPLICA	mongo:40	1	1	FARGATE	LATEST(1.4.0)
serve-ipsec-daemon	ACTIVE	REPLICA	serve-ipsec-da...	1	1	FARGATE	LATEST(1.4.0)
app	ACTIVE	REPLICA	app:22	1	1	FARGATE	LATEST(1.4.0)
serve-openvpn-udp	ACTIVE	REPLICA	serve-openvpn...	1	1	FARGATE	LATEST(1.4.0)
frontend	ACTIVE	REPLICA	frontend:16	1	1	FARGATE	LATEST(1.4.0)
event-server	ACTIVE	REPLICA	event-server:10	1	1	FARGATE	LATEST(1.4.0)
mailcatcher	ACTIVE	REPLICA	mailcatcher:8	1	1	FARGATE	LATEST(1.4.0)
goredns	ACTIVE	REPLICA	goredns:11	1	1	FARGATE	LATEST(1.4.0)
ocsp-responder	ACTIVE	REPLICA	ocsp-responde...	1	1	FARGATE	LATEST(1.4.0)
serve-openvpn-tcp	ACTIVE	REPLICA	serve-openvpn...	1	1	FARGATE	LATEST(1.4.0)
samba-ad-dc	ACTIVE	REPLICA	samba-ad-dc:3	1	1	FARGATE	LATEST(1.4.0)
provision	ACTIVE	REPLICA	provision:7	1	1	FARGATE	LATEST(1.4.0)

Figure 8: Certidude services (Source: author created)

Containers are then built locally and uploaded them to container registry ECR using AWS CLI.

There are 2 major models for running containers in AWS - Fargate and EC2. EC2 is for self-managing containers on EC2 and Fargate is for running them in serverless fashion. With Fargate, we can also achieve low overhead and the price is based on CPU and memory usage. Since this isn't meant to be for production, Fargate is a good fit for our use case [6]. Logs can't be viewed directly if using Fargate so we need to configure awslogs in task definition to view logs through Cloudwatch.

The only networking option in Fargate is awsvpc, which gives tasks its own elastic network interface (ENI) and a primary private IPv4 address like networking properties of EC2 [9].

For containers in different services to talk to each other, they need to have Service Discovery enabled. AWS ECS Service Discovery uses CloudMap API to manage HTTP and DNS namespaces for containers [14].

Once run, Fargate service can't be stopped directly because they are not relying on EC2 that customers can control. Fargate is meant to run containers in a serverless fashion so that users don't have to take care of the underlying resources. The only way to stop Fargate service is to change the desired running task in service to 0. Here is a small bash script to start and stop Certidude Fargate services with AWS CLI:

```
_certidude_f() {
        for service in $(aws ecs list-services --cluster Certidude | jq
-r '.serviceArns[]' | grep -o '[a-z-]*$'); do
                if aws ecs update-service --cluster Certidude --service
$service --desired-count "$1" > /dev/null; then
                        echo "Success: ${service}"
                else
                        echo "Failure: ${service}"
                fi
        done
}

alias certidude_down="_certidude_f 0"
alias certidude_up="_certidude_f 1"
```

Figure 9: Fargate automation script (Source: author created)

Security group acts as a virtual firewall for instance to control inbound and outbound traffic. Since security group acts as instance level, not in subnet level, being in the same security group only means that they share the same rules, not that they could talk with each other.

Autoscaling service is used to make sure that services stay online when traffic increases unexpectedly.

Option name *restart* is not supported for EC2 task definition but this is not a problem.

After researching, it's concluded that Fargate is not useful for Certidude because it doesn't support essential parameters such as privileged and devices, which are crucial for Certidude as it needs to run multiple OpenVPN daemons. In this case, load balancing between daemons could hardly be done. The answer for this might be HAProxy TCP method but it won't work with UDP. For IPSec, it uses whole different protocol ESP.

As a result, from 12 task definitions is moved back to only 1 which contains all of the necessary containers, and replicate them along with demand. This also removes ServiceDiscovery aspect because now all containers are in the same task.

Container Definitions

Container Name	Image	CPU Units	GPU	Inference Acceler...	Hard/Soft memory limits (MiB)	Essential
server-backend	122358940018.dkr....	0			--/--	true
server-events	122358940018.dkr....	0			--/--	true
server-frontend	122358940018.dkr....	0			--/--	true
server-goredns	122358940018.dkr....	0			--/--	true
server-ocsp-responder	122358940018.dkr....	0			--/--	true
server-openvpn-tcp	122358940018.dkr....	0			--/--	true
server-openvpn-udp	122358940018.dkr....	0			--/--	true
server-provision	122358940018.dkr....	0			--/--	false
server-strongswan	122358940018.dkr....	0			--/--	true

Figure 10: Certidude task definition containers (Source: author created)

Because now all the containers are running in the same instance, t3 micro is not enough so moving from t3 micro to t3 medium. In addition to launching a t3 medium instance, adding these lines to user data for ECS to detect the instance:

```
#!/bin/bash
echo "ECS_CLUSTER=Certidude" >> /etc/ecs/ecs.config
```

Figure 11: Adding to user data (Source: author created)

4.2 Implementation issues

Docker compose files couldn’t be used as templates to deploy containers to AWS because AWS currently only supports major docker-compose versions such as 3.0, 2.0 so for Certidude AWS docker-compose, the version is changed from 3.7 to 3.0.

Container-transform is a third-party tool and still has limitations. For example, some requirements still have to be manually specified such as log driver, container images, memory, cpu, volumes, etc.

ECS limits to 10 containers per task definition so for application like Certidude, which has about 11 containers that share data volumes, bind-mount option for data volumes is not a solution because 11 containers couldn’t be put in the same task definition.

At the time of deployment, there was a problem that developers update image names frequently. Since task definition was generated and uploaded containers remotely, this was a problem for me to keep up with deploying process. Therefore, another script is written for getting container names, pushing image to ECR and automatically creating new ones if it wasn’t created (because of the name changing).

```
_get_service_names() {
                aws  ecs  list-services  --cluster  Certidude  |  jq  -r
'.serviceArns[]' | grep -o '[a-z-]*$'
}

_get_container_names() {
         aws ecs describe-task-definition --task-definition certidude |
jq -r '.taskDefinition.containerDefinitions[].name'
}

certidude_image_push() {

        local arn='example.dkr.ecr.eu-north-1.amazonaws.com'
         aws ecr get-login-password --region eu-north-1 | docker login
--username AWS --password-stdin "$arn"
        for service in $(_get_container_names); do
                local tag="certidude-mongodb-migration_${service}"
                local remote_tag="${arn}/${tag}:latest"
                  aws ecr create-repository --repository-name "$tag" --
region eu-north-1  || { echo "Failed to create repository for ${tag}";
break; }
                   docker tag "$tag" "$remote_tag" || { echo "Missing
container: ${tag}"; break; }
                  docker push "$remote_tag" || { echo "Failed to push $
{remote_tag}"; break; }
        done

}
```

Figure 12: Pushing image automation script (Source: author created)

5 Summary

5.1 Conclusion

In conclusion, following options are recommended for Certidude:

- ECS EC2 as the only option for running containers.
- For database, it's recommended that organizations choose to use DocumentDB.
- Secrets Manager for Privacy Key rotation.
- Route53 for hosted zone.
- S3 and CloudFront for front-end.
- Lambda will be tested for server and client API.
- ELB should be used for region replication.
- CloudHSM for backing keys.

Organizations might worry about deploying such a critical part of their infrastructure to the cloud, but there is still HSM support for secure key storage/usage [25].

There are many generic CA services available from third-party providers on AWS MarketPlace for example. However, what these are bad about is enforcing best practice.

To simplify current paper, some functionalities are not specifically discussed due to their complexity such as routing of the traffic inside VPN, routing protocols, NAT, iptables.

5.2 Further work

This project will be continued with implementing this architecture. Some of the ideas are writing CloudFormation template because as an infrastructure-as-code tool it will bring many benefits such as scalability, visibility, stability, and security [7].

At the current time, Jenkins is already implemented as a development pipeline for Certidude. Connect Jenkins with AWS will help with pushing container images so no need to use the script.

Last but not least, all of the mentioned security enhancement suggestions will be investigated and tested to ensure Certidude security on the cloud.

References

[1] Acloud-Guru, "AWS Certified Solution Architect Professional", 2018. [Online]. Available: https://acloud.guru/forums/aws-certified-solutions-architect-professional/discussion/-L9D8QP7KMoh8Mvg2T2q/AWS%20KMS%20vs%20CloudHSM%3F [Accessed 5 May 2021].

[2] Amazon Official Blog, "Deploy applications on Amazon ECS using Docker Compose", 2020 [Online]. Available: https://aws.amazon.com/blogs/containers/deploy-applications-on-amazon-ecs-using-docker-compose/ [Accessed 5 May 2021].

[3] Amazon Official Blog, "Migrating Applications from Monolithic to Microservice on AWS", 2019 [Online]. Available: https://aws.amazon.com/blogs/apn/migrating-applications-from-monolithic-to-microservice-on-aws/ [Accessed 5 May 2021].

[4] Amazon Official Blog, "The 5 Pillars of the AWS Well-Architected Framework", 2018 [Online]. Available: https://aws.amazon.com/blogs/apn/the-5-pillars-of-the-aws-well-architected-framework/ [Accessed 5 May 2021].

[5] Amazon Official Container Guide. "Architecture Patterns", 2018 [Online]. Available: https://containersonaws.com/architecture/ [Accessed 5 May 2021].

[6] Amazon Official Container Guide, "EC2 or AWS Fargate?", 2018 [Online]. Available: https://containersonaws.com/introduction/ec2-or-aws-fargate/ [Accessed 5 May 2021].

[7] Amazon Official Container Guide, "Why use infrastructure as code?", 2018 [Online]. Available:ex https://containersonaws.com/introduction/infrastructure-as-code/ [Accessed 5 May 2021].

[8] AWS Official Container Guide. "Autoscaling Services", 2018 [Online]. Available: https://containersonaws.com/architecture/autoscaling-service-containers/ [Accessed 5 May 2021].

[9] Amazon Official Website, "Deploy applications on Amazon ECS using Docker Compose", 2020 [Online]. Available: https://aws.amazon.com/blogs/containers/deploy-applications-on-amazon-ecs-using-docker-compose/ [Accessed 5 May 2021].

[10] Amazon Official Website, "AWS Fargate Pricing", 2021 [Online]. Available: https://aws.amazon.com/fargate/pricing/ [Accessed 5 May 2021].

[11] Amazon Official Website, “VPC Security Groups”, 2021 [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html [Accessed 5 May 2021].

[12] AWS Official Website, “App Mesh”, 2021 [Online]. Available: https://aws.amazon.com/app-mesh/ [Accessed 5 May 2021].

[13] Amazon Official Website, “Fargate Task Networking”, 2021 [Online]. Available: https://docs.aws.amazon.com/AmazonECS/latest/userguide/fargate-task-networking.html [Accessed 5 May 2021].

[14] Amazon Official Website, “Service Discovery”, 2021 [Online]. Available: https://docs.aws.amazon.com/AmazonECS/latest/developerguide/service-discovery.html [Accessed 5 May 2021].

[15] AWS Official Website, “Using Spot Instances”, 2021 [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances.html [Accessed 5 May 2021].

[16] AWS Official Website, “Fargate Task Definitions”, 2021 [Online]. Available: https://docs.aws.amazon.com/AmazonECS/latest/userguide/fargate-task-defs.html [Accessed 5 May 2021].

[17] AWS Official Website, “Reserved Instances”, 2021 [Online]. Available: https://aws.amazon.com/ec2/pricing/reserved-instances [Accessed 5 May 2021].

[18] AWS Official Website, “Saving Plans”, 2021 [Online]. Available: https://aws.amazon.com/savingsplans/ [Accessed 5 May 2021].

[19] AWS Official Website, “CloudHSM”, 2021 [Online]. Available: https://aws.amazon.com/cloudhsm/ [Accessed 5 May 2021].

[20] AWS Official Website, “KMS”, 2021 [Online]. Available: https://aws.amazon.com/kms/ [Accessed 5 May 2021].

[21] AWS Official Website, “Microsoft PKI”, 2021 [Online]. Available: https://aws.amazon.com/quickstart/architecture/microsoft-pki/ [Accessed 5 May 2021].

[22] Amazon Official IAM Guide, “AWS Best Practices”, 2021 [Online]. Available: https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#lock-away-credentials [Accessed 5 May 2021].

[23] AWS Whitepaper, “Microservices on AWS”, 2021 [Online]. Available: https://d1.awsstatic.com/whitepapers/microservices-on-aws.pdf [Accessed 5 May 2021].

[24] Casey Gibson , “Difference between AWS DynamoDB and AWS DocumentDB and MongoDB”, 2019 [Online]. Available: https://medium.com/@caseygibson_42696/difference-between-aws-dynamodb-vs-aws-documentdb-vs-mongodb-9cb026a94767 [Accessed 5 May 2021].

[25] Cloudonaut, “Review AWS AppMesh, EC2, ECS, EKS”, 2020 [Online]. Available: https://cloudonaut.io/review-aws-app-mesh-service-mesh-ec2-ecs-eks/ [Accessed 5 May 2021].

[26] Cloudonaut , “ECS vs Fargate What is the Difference”, 2019 [Online]. Available: https://cloudonaut.io/ecs-vs-fargate-whats-the-difference/ [Accessed 5 May 2021].

[27] Certidude Github Page [Online]. Available: https://github.com/laurivosandi/certidude [Accessed 5 May 2021].

[28] DB-engines, “Comparision between DynamoDB and MongoDB”, 2021 [Online]. Available: https://db-engines.com/en/system/Amazon+DynamoDB%3BMongoDB [Accessed 5 May 2021].

[29] Edureka, “What is the difference between Amazon ECS and Amazon EC2”, 2018 [Online]. Available: https://www.edureka.co/community/9549/what-is-the-difference-between-amazon-ecs-and-amazon-ec2 [Accessed 5 May 2021].

[30] Empathy.co, “Build a cheaper, more flexible VPN solution on AWS with our open-source OpenVPN Certificate Authority”, 2019 [Online]. Available: https://medium.com/empathyco/build-a-cheaper-more-flexible-vpn-solution-on-aws-with-our-open-source-openvpn-certificate-1a94661ac0af#8278 [Accessed 5 May 2021].

[31] Stackoverflow, “What is the difference between a task and a service in AWS ECS?”, 2017 [Online]. Available: https://stackoverflow.com/questions/42960678/what-is-the-difference-between-a-task-and-a-service-in-aws-ecs [Accessed 5 May 2021].

[32] Micah Hausler, Container Transform Github, 2017 [Online]. Available: https://github.com/micahhausler/container-transform [Accessed 5 May 2021].

[33] PrimeKey, “EJBCA Cloud”, 2021 [Online]. Available: https://www.primekey.com/products/cloud/ejbca-cloud/ [Accessed 5 May 2021].

[34] PrimeKey, “Why Would You Deploy Your PKI In The Cloud”, 2018 [Online]. Available: https://www.primekey.com/blog/2018/11/29/why-would-you-deploy-your-pki-in-the-cloud/ [Accessed 5 May 2021].

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis[1]

I Song Huong Pham Thi

1 Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis “Converting Legacy Application to the Cloud: The case of Certidude”, supervised by Lauri Võsandi and Toomas Lepikult.

1.1 to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;

1.2 to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.

2 I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.

3 I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

29.04.2021

1 The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 – Docker compose file for AWS task defintion generation

```
Version: '3.0'
volumes:
    authority-secrets:
    server-secrets:
    mongodb:
    vpn-configs:
    samba-secrets:
    client-strongswan:
    client-openvpn:
    client-shield:
services:

  # Main backend
  server-backend:
     image: example.dkr.ecr.eu-north-1.amazonaws.com/certidude-mongodb-
migration_server-backend
    restart: always
    volumes:
        - authority-secrets:/var/lib/certidude/authority-secrets:ro
        - server-secrets:/var/lib/certidude/server-secrets:ro
        - samba-secrets:/var/lib/samba/private/tls:ro
        - /dev/random:/dev/random:ro
        - /tmp/coverage:/tmp/coverage
     command: pinecone serve backend
     environment: arn:aws:s3:::certidude/common.env

  # Nginx serves the 80, 443, 8443 entry points
  server-frontend:
     image: example.dkr.ecr.eu-north-1.amazonaws.com/certidude-mongodb-
migration_server-frontend
    restart: always
```

```
    volumes:
     - server-secrets:/var/lib/certidude/server-secrets:ro

  # OCSP responder
  server-ocsp-responder:
     image: example.dkr.ecr.eu-north-1.amazonaws.com/certidude-mongodb-
migration_server-ocsp-responder
    restart: always
    volumes:
     - ./pinecrypt:/src/pinecrypt:ro
     - authority-secrets:/var/lib/certidude/authority-secrets:ro
     - server-secrets:/var/lib/certidude/server-secrets:ro
     - /tmp/coverage:/tmp/coverage
  server-events:
     image: example.dkr.ecr.eu-north-1.amazonaws.com/certidude-mongodb-
migration_server-events
    restart: always
    volumes:
     - samba-secrets:/var/lib/samba/private/tls:ro
     - /tmp/coverage:/tmp/coverage
    command: pinecone serve events

  # Provision keys
  server-provision:
     image: example.dkr.ecr.eu-north-1.amazonaws.com/certidude-mongodb-
migration_server-provision
    command: pinecone provision
    volumes:
     - server-secrets:/var/lib/certidude/server-secrets
     - authority-secrets:/var/lib/certidude/authority-secrets
- /dev/random:/dev/random:ro
    image:  example.dkr.ecr.eu-north-1.amazonaws.com/certidude-mongodb-
migration_server-openvpn-udp
    command: pinecone serve openvpn --client-subnet-slot 0
    cap_add:
     - NET_ADMIN
    volumes:
     - server-secrets:/var/lib/certidude/server-secrets:ro
     - /tmp/coverage:/tmp/coverage
    devices:
     - /dev/net/tun:/dev/net/tun

  # Serve OpenVPN gateway via TCP 443
  server-openvpn-tcp:
    restart: always
     image: example.dkr.ecr.eu-north-1.amazonaws.com/certidude-mongodb-
migration_server-openvpn-tcp
    command: pinecone serve openvpn --client-subnet-slot 1 --proto tcp
```

```
    cap_add:
     - NET_ADMIN
    volumes:
     - server-secrets:/var/lib/certidude/server-secrets:ro
     - /tmp/coverage:/tmp/coverage
    devices:
     - /dev/net/tun:/dev/net/tun


  server-goredns:
     image: example.dkr.ecr.eu-north-1.amazonaws.com/certidude-mongodb-
migration_server-goredns

  server-strongswan:
     image: example.dkr.ecr.eu-north-1.amazonaws.com/certidude-mongodb-
migration_server-strongswan
     restart: always

  # Serve OpenVPN gateway via UDP 1194
  server-openvpn-udp:
    restart: always
     command: pinecone serve strongswan --client-subnet-slot 2
     cap_add:
      - NET_ADMIN
     volumes:
      - server-secrets:/var/lib/certidude/server-secrets:ro
      - /tmp/coverage:/tmp/coverage
     devices:
      - /dev/net/tun:/dev/net/tun
```

Appendix 3 – AWS CLI

```
ecs-cli compose --project-name certidude --file docker-compose.aws.yml
--ecs-params ecs-params.yml create

aws ecs describe-task-definition --task-definition certidude
```