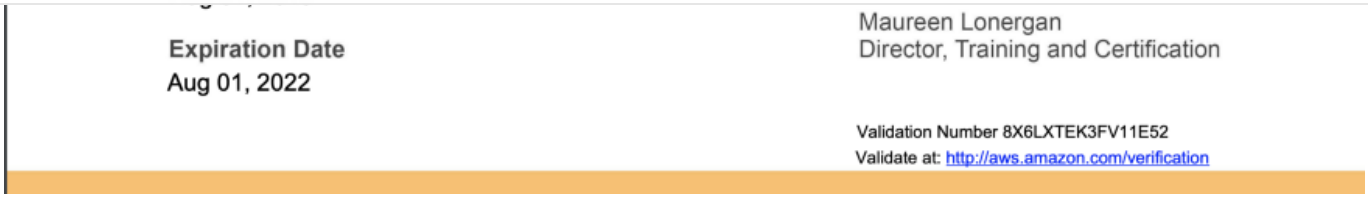
Aram Petrosyan

Follow 139 Followers About

What it takes to pass the AWS Certified DevOps Engineer — Professional exam

Aram Petrosyan Aug 17, 2019 · 16 min read

Having completed AWS Certified DevOps Engineer — Professional Certification exam after the second attempt with 90% score, I thought it would be useful to share my learning path and the mistakes I made the first time so you can avoid them.



> *To successfully pass the exam you must know* ***lots of things about many things.***

A lot has been said about the professional exam in blog posts and forums, many training videos are available online like AWS Certified DevOps Engineer — Professional 2019 or AWS Certified DevOps Engineer — Professional Level to study (the last one worked better for me). They all are amazing resources to make use of for studying and to get good tips&tricks. So go ahead watch and read them all, very important to do all the hands-on they provide. **They are necessary** to take, at least one of these courses, to prepare you but not enough to successfully complete the exam. That's what I did myself and it wasn't enough. So I guess what I am trying to share here is how to pick up the missing parts.

Having taken the exam twice I noticed that the exam is designed to test not only your knowledge of cloud services available on AWS, your ability to design and create a highly available, scalable, fault-tolerant and self-healing system, but it is also meant to test how native you are to the cloud. What I mean by this, is the exam is long and tiring (3 to 3.5 hours) with 75 scenario-based and problem-based questions with long answers which all seem to be correct. After 35–40 questions your brain starts melting down and it hurts when you try to think and analyse, leaving you with the only option of using your intuition to eliminate out the answers which are least correct until you are left with the remaining correct answer which satisfies all the conditions of the question.

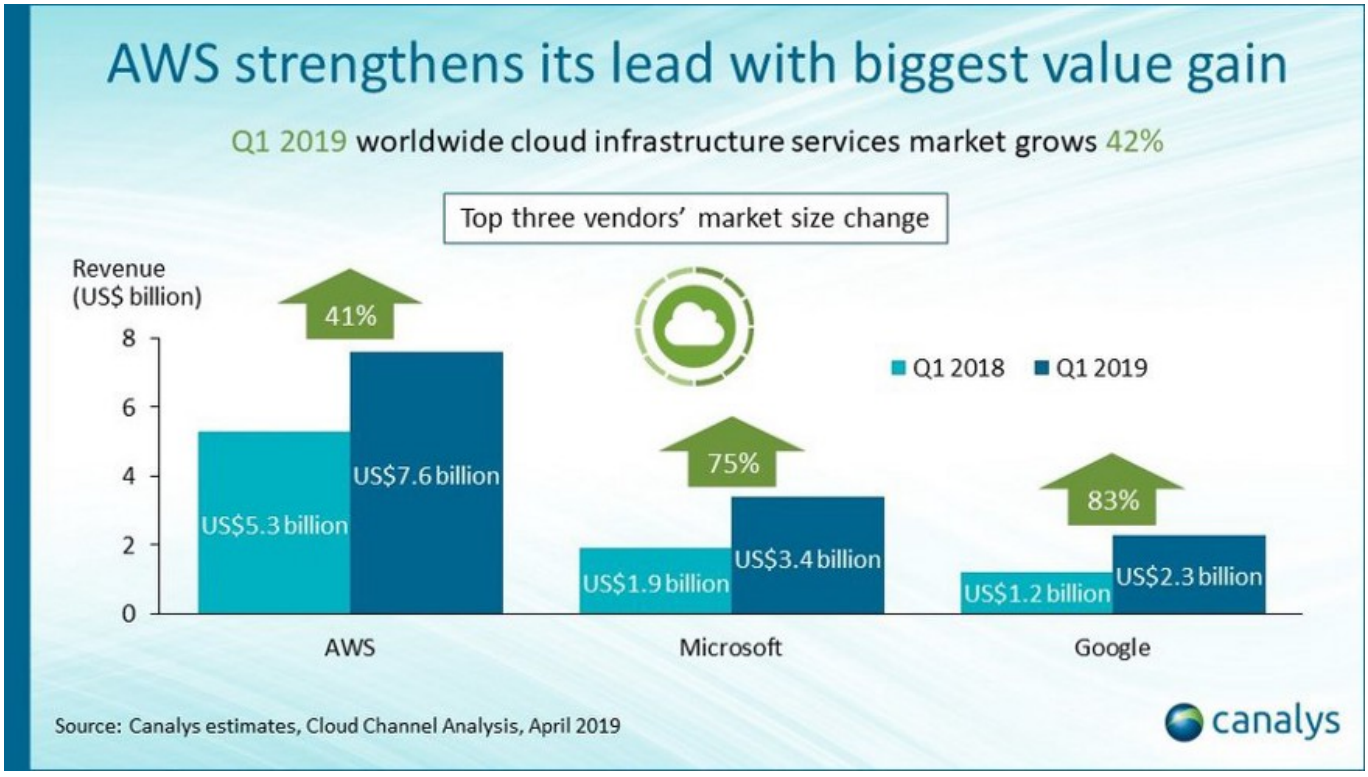
In order to do that you have to be familiar with AWS like a fish in the ocean, basically a cloud-native. Even if you have experience working with AWS, even if it is 2 years+ in many cases, you don't get the chance to work with all the services you need to know to

So my strategy was to take a simple *(almost)* hello-world app and make it multi-region self-healing with disaster recovery readiness. You are going to need two AWS accounts and expect approximately a $100 bill after all the experiments you are going to do. I recommend that you create two new accounts even if you have an existing one. A new account has lots of free tiers.

Notice: This is also not a complete guide on how to pass the exam. Search for other blog posts on the Internet to see what others say, watch re:invent 300 and 400 videos — they are great, read AWS DevOps blog posts, listen to AWS podcasts on Spotify and use all other available resources.

Motivation

Amazon Web Services is one of the biggest names in the IT industry and is leading in cloud computing. According to the market share report AWS is the leader leaving others far behind.

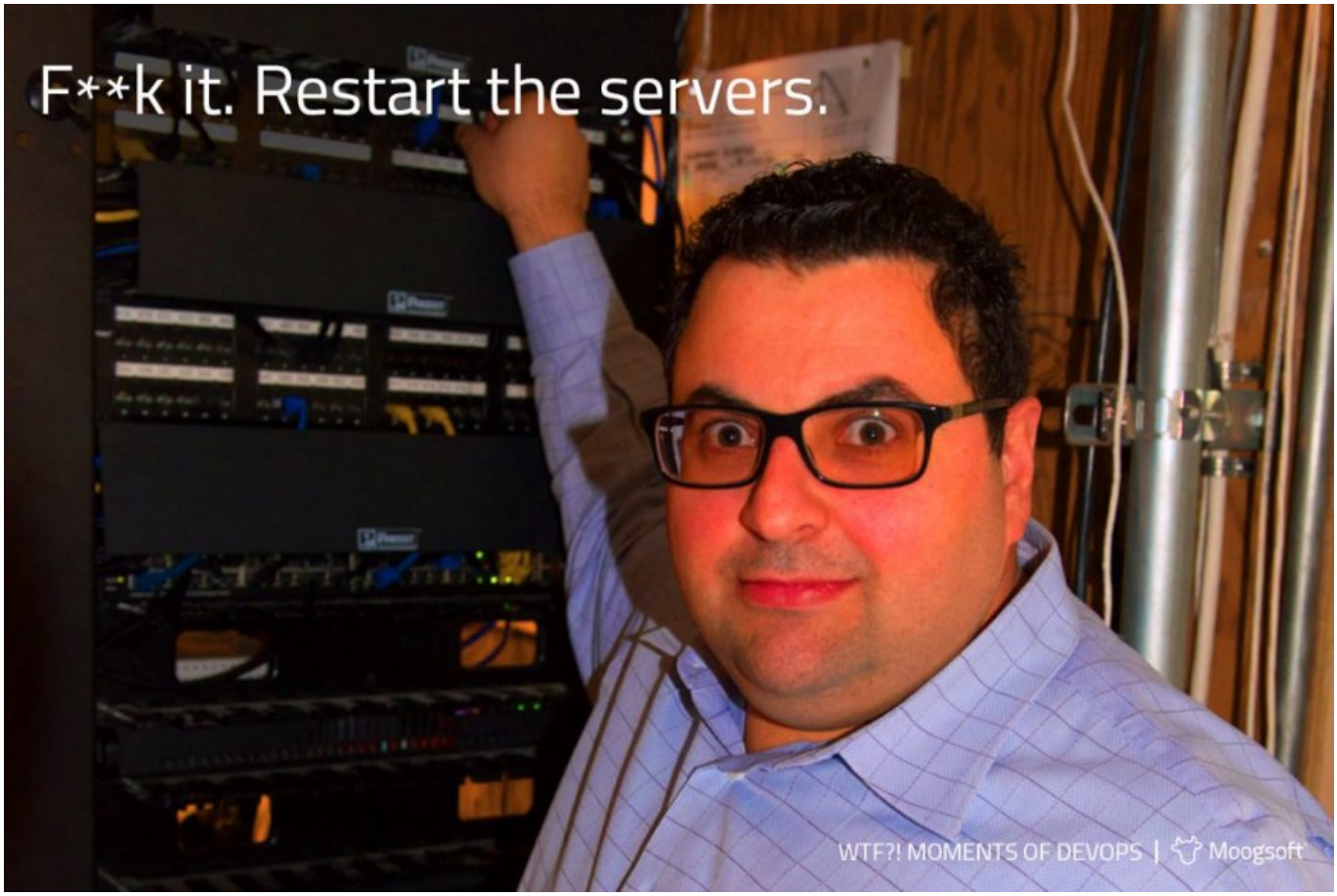


out of it to satisfy all requirements, compliance and security policies of the company.

Many things have also been said about the benefits and I will just add one which hasn't been mentioned. While you are preparing for the exam you learn, afterwards you will be left with an enormous amount of knowledge, knowledge of how Amazon engineers solve problems, what good engineering looks like, a good engineering ethic, how to design documentable solutions and how to document your code so other engineers can use it.

> *It is not just a digital piece of paper to ask more salary or boost your self-esteem, it is a good investment in your career, growth and in yourself.*

What is DevOps?



Before you can even attempt to take the DevOps exam, you have to clarify for yourself what DevOps is and why it exists? Leaving the official and formal definitions behind,

make the thing work **on remote machines.**

Avoiding the question who is a DevOps (if there is a role DevOps 👧👧) it is important to understand that first of all, DevOps is a culture where you as an engineer can develop a software with CI/CD pipeline, make sure it works not only on **your machine** but also on production, you know how to monitor it, read logs, debug, support, fire-fight on production, keep track of all company's convention on naming, sizing and resource usage. In other words, it is taking ownership right from the beginning to the delivery to the end-user.

It is also a good idea to take time and read AWS understanding of DevOps. It will give you more insight into what you are going to be tested on.

The fun... 🤓

A few pre-requirements

1. Leave the laziness here and spend as much time as you can
2. Register two AWS accounts
3. You will only use CloudFormation and AWS CLI but never AWS Management Console to create/modify the infrastructure. **This is very important.** If you use Management Console you will miss half of the important options you need to know. Management Console takes care of creating all dependencies by one tick of a checkbox and setting default values for required options. Even though it makes sense to create infrastructure using the Management Console to see how things will look like if you don't have much experience and then burn it after and do the same using CloudFormation and CLI. I can't **emphasize enough** how critical this is to understand how services are connected to each other and how they communicate
4. This help-guide is not about how to do things rather it is more about what to do to get ready for the exam. So I will provide tasks I did solve step by step and you can try to solve them by yourself

6. Presumably, at this point, you are at a good level of knowledge of `EC2` instances, `ASG` and `ALB` s and `CLB` s, a good level of understanding CI/CD pipelines and different deployment strategies and can decode all abbreviations. If not please go back and take one of the lessons mentioned earlier in this post

7. If you are not very confident at using CloudFormation probably it is better to start off by taking AWS Advanced CloudFormation course. This will give you confidence and deep understanding of IaaC

Preparation

We are going to use Go because it is a compilable language with easy dependency management and has a built-in east-to-run web server. Grab the code below

```
1  package main
2
3  import (
4          "flag"
5          "fmt"
6          "os"
7          "net/http"
8
9          "github.com/aws/aws-lambda-go/events"
10         "github.com/aws/aws-lambda-go/lambda"
11 )
12
13 func main() {
14         mode := flag.String("mode", "", "server mode")
15
16         flag.Parse()
17
18         fmt.Println(*mode)
19         if *mode == "web" {
20                 fmt.Println("Start web")
21                 startWeb()
22         } else {
23                 fmt.Println("Start lambda")
24                 startLambda()
25         }
```

```
29          lambda.Start(func(request events.APIGatewayProxyRequest) (events.APIGatewayProxyResponse
30                  return events.APIGatewayProxyResponse{
31                          StatusCode: http.StatusOK,
32                          Body: getContent(),
33                  }, nil
34          })
35  }
36
37  func startWeb() {
38          port := os.Getenv("port")
39          if port == "" {
40                  port = "8080"
41          }
42
43          http.HandleFunc("/", func(writer http.ResponseWriter, request *http.Request) {
44                  switch request.URL.Path {
45                  default:
46                          writer.WriteHeader(http.StatusNotFound)
47                  case "/":
48                          _, _ = fmt.Fprint(writer, getContent())
49                  case "/health":
50                          writer.WriteHeader(http.StatusOK)
51                          _, _ = fmt.Fprint(writer, "healthy")
52                  }
53          })
54          _ = http.ListenAndServe(":" + port, nil)
55  }
56
57  func getContent() string {
58          return "Hello, DevOps!"
59  }
```

If you are not familiar with Go here is a quick instruction on how to launch and stop the app

```
# install dependancies
```

```
# build the app

go build -o app main.go
chmod +x app

# launch the app

## Lambda mode

./app

## Web Server mode

./app -web

# stop the app

ps ax | grep app | grep -v grep | awk '{print $1}' | xargs kill
```

instructions.md hosted with ♡ by GitHub view raw

Dodgy but does the job!

Create a new user on one of the AWS accounts, lock up the root account and turn on MFA for the root account and all users. **This is an important security cause.**

A standalone fleet of EC2 instances

1. Launch 5 `t2.micro` EC2 instances

would normally do on GitHub or other known Git Repository Services. See what are the other ways you can use to access code commit, thought HTTPS or using AWS credentials. How people would access the repository from a different account if they can

3. Create a CodePipeline using AWS Code services to build and deploy the sample app to the newly launched instances

Find out how you can target EC2 instances you want to deploy on, how to set up a pipeline, how the pipeline transfers `Artefacts` between stages, how `Artefacts` are encrypted at rest

1. Try out different deployment strategies `AllAtOnce` `HalfAtOne` `OneAtOnce` for `in-place` deployment
2. Try to create a custom deployment configuration for `in-place` deployment and deploy with custom configuration
3. See if you can do a `Blue/Green` deployment. If not, understand why?!

Next, launch a new `ALB` and register the instances with appropriate health check endpoint, block any traffic to instances and allow only access for `ALB` and try out

1. `Blue/Green` deployment and see how it goes. Find out how to target `EC2` instances
2. Try `Blue/Green` deployment with different deployment configuration `AllAtOnce` `HalfAtOne` `OneAtOnce` see if there is any difference
3. See how traffic draining works

This will give you a good understanding of how AWS Code services work, how they communicate with each other, how they store `Artefacts`

Rollback

stop and rollback the deployment automatically if say `500` errors count hit a threshold. Find a way to solve the problem. Run another 15 `t2.micro` `EC2` instances and deploy the code on them as well.

After successfully launching the new instances with running application change the code to response `500` on the main `/` endpoint and leave `200` for `/health` endpoint. Come up with your own solution and commit the code, meanwhile, grab `ALB` hostname after the deployment process has started in the pipeline and keep refreshing in a browser until you hit your defined threshold and see if your solution worked.

Also, think how you would stop and rollback automatically based on application logs which may output some internal failures e.g. amount failed transactions or number of failed processed messages in a queue. Add some logging in the app which will randomly log `success` or `failure` and try to rollback based on that logs

You will be tested on these topics **very thoroughly**

Autoscaling Group ♡

Now, after you successfully experimented CI/CD on standalone fleet it is time to think about high-availability of the sample app. One of the flagships of AWS high-availability is `ASG`

Create a launch configuration or template and run a new `ASG` with minimum 2 and maximum of 20 instances set the desired capacity to 5. Setup a scheduled action to increase the desired capacity. Let's assume people are more eager to see our hello world app after working hours. So let's get the desired capacity to 20 after 6 PM o'clock and roll back to 5 at 11.00 PM.

1. Register 20 running instances to `ASG` see what happens, how AZBalancing works with traffic draining
2. Change your pipeline to deploy using `ASG` , do the same steps with deployment and rollback but target `ASG` . See if there are any differences, understand how

3. Figure out what will happen if a scale-out gets triggered during a deployment process or rollback. Would the new instances be launched with the new code or the old and why. It is critically important to understand how `CodeDeploy` and `ASG` coordinate processes and how they handle edge cases
4. Try to suspend different processes of `ASG` and see how they affect the deployment. Turn them on and off to experiment
5. What will happen with `StandBy` instances on a new deployment or in case of a scale-out event
6. Try to stop the running app before the instance gets terminated. You will need to be able to run commands on instances before their termination or stop running application gracefully

Use stress to stimulate CPU stress load in your instances to artificially generate scale-out/scale-in events.

If you don't have answers to these questions don't even attempt to take the exam, save you some $360 😉

Elastic Beanstalk 😫

I know the feeling! But this is also a must. After you cried out let's make sure of cross-region availability of our simple app

1. Change the region and launch a new Elastic Beanstalk application with an high-available environment
2. Go back to the original region and modify Code Pipeline to get the simple app deployed on Elastic Beanstalk in the new region. Grab environment's URL and make sure it is working on a browser
3. By default, Elastic Beanstalk configures `ASG` to use `NetworkIn` . Try to change it to `CPUUtilization` as a part of the deployment process

5. Change the `ASG` scale-out option back yo `NetworkIn` using CLI and how precedence works in Elastic Bean Stalk. Which one of `.ebextensions` `EB CLI` `Configuration` has higher priority and why
6. Try out different deployment strategies for Elastic Beanstalk. See how you would implement `Blue/Green` deployment and if you can do it using `CodePipeline`
7. Understand the difference between `rolling` and `immutable` updates. Pros and cons of each one

Pre-baked AMI

Create an AMI from one of the running `EC2` and make sure all the containers run on the newly backed AMI. **It should be automatically and never a manual step**.

Also, find a way for cross-region and cross-account distribution of the AMI in case of DR

Have a read about Golden AMI Pipeline and understand and remember the architecture of baking AMI as scheduled action, manually launch and CI/CD. Try to replicate it on your account

Elastic Container Service

Photo by frank mckenna on Unsplash

Time to Docker ♡

1. Add another build stage in your Pipeline which packs the code and dependencies into a Docker Image and push it to `ECR`
2. Create a new `ECS` cluster in a new region and figure out how to deploy using `CodeDeploy` .
3. Add another stage in the Code Pipeline in the original region to deploy the app on the cluster in the new region
4. Try different deployment techniques `in-place` `blue/green` `rolling`
5. See if you can also set up a new deployment stage which will deploy to Elastic Beanstalk Docker environment
6. See if you can have `ECR` cross-account access

There are quite a few questions of dockerization and containerization

API Gateway

`Serverless` is not widely covered by the exam but sill it is covered so make sure you know how to serverless.

1. Create a SAM template
2. Add a new deployment stage on your pipeline which deploys the code on an entirely new region using `CodeDeploy`
3. Find out how to set up a canary deployment with `CodeDeploy` and how preTrafficHooks work

5. Observe the traffic shifting process between versions
6. After successfully completing try out deploying by using `CodePipeline` and `Cloudformation`. See what are the differences, which one is more preferable in which situations.

Database

Data storage, data replication, and high availability of the data is also a huge chunk and well-covered in the exam. So let's get some database involved in the experiment. Create a new RDS whichever database engine you fancy. Here is the list of supported SQL Driver for GoLang

Try to add a new endpoint like `/store` to store some data into the database

1. In `KMS` create a new `CMK`. It is also important to understand how `KMS` permissions and policies work
2. Add database credentials to `SSM` as `SecretScring` using the `CMK`
3. Modify the role of running instances, lambda functions, ECS tasks so it has access to get parameters from `SSM` and also it is allowed to decrypt the encrypted string by `CMK`
4. Use the endpoint to store some data into the table
5. Set up read replicas
6. See if you can set up multi-az, multi-region, and multi-account replication
7. Set up scheduled backups and distribute the backup to other region and the second account
8. Think a way to backup `SSM` parameters and distribute to other regions or on the second account in case of DR
9. See all the events RDS triggers to `Cloudwatch`

centre going down. So multi-account replication of the data is kind of a big deal if the DR is critical. Depending on `RTO` and `RPO` you should be able to launch your infrastructure in a new AWS account (in an ideal world) with the data

Route53

Now, so far you should have the sample app deployed on different computing engines in different regions.

First things first register the cheapest domain you can find on AWS.

1. Set up traffic distribution so that `50%` of the traffic goes to the main region and the other `50%` distributed evenly between regions
2. Set up failover and health checks so that when you tear apart one region it will automatically reroute the traffic to the healthy region
3. Host a static website on `S3` which is the last failover with some static `oops` content

A/B testing

A/B testing is one of the important components of the modern application shipment ecosystem. Imagine you want to split the application traffic not only by random percentage but based on data that a request contains like a browser type, IP address, cookies, query-string, etc…

Configure an A/B testing on `Route53` which makes decisions based on a query-string. So if the query string has

1. `?type=1` redirect traffic to the `ASG` created in the first region
2. `?type=2` redirect traffic to `Elastic BeanStalk`
3. `?type=3` to `ECS`
4. etc…

and `Firefox` traffic to `Elastic Beanstalk`

Config & administration

Finally, when you have created a decent amount of resources it is time to think about compliance and configuration. Make sure all your running instances have a tag `purpose = DevOps` exam, all your `s3` buckets are encrypted by default, no role has administrator access. The most important part is it all should be automated. No manual step. In the end, if you launch a new `EC2` container which is not compliant should be addressed and automatically fixed

CAUTION: When you apply AWS Config rule it charges you $1 instantly, so be thoughtful and don't apply all available rules at once.

Additionally, install the `AWS Inspector` agent on all running instances and see how `AWS Inspector` works and what's the way to automatically address all notifications `AWS Inspector` generates

> *These instructions may or may not contain distructive statments. Some of them may or may not be possible to achive. It is up to you to figure out and make decitions. That's what you will be doing 3 hours in the exam* 🦉

But, in the end, you should have the simple app deployed in 5–6 different regions on different computing engines which is highly available, A/B testable, disaster recovery ready, auto-scales to handle any traffic. If you tear down region one by one it should automatically failover and eventually show you `oops` static webpage hosted on `s3`. if you kill the main database instances it also should automatically do AZ failover or able to be spun up in a new region within minutes and lastly, if you remove AWS account you should be able to run the same infrastructure on the other account within a couple of hours with recovered data. This will include automatic replication of the `s3` data to the second account.

You will also end up having multiple CloudFormation files as I did, which, obviously, I am not going to share. You are all lazy butts like I am would copy-paste like I would do

If you did all this yourself I am pretty sure you are ready to schedule your exam and smash it 💪

What's next?

If you don't want to get a score like this after your exam, please read carefully below.

aws training and certification

AWS Certified DevOps Engineer - Professional

Notice of Exam Results

Candidate: Aram Petrosyan	Exam Date: Jul 03, 2019
Candidate ID: AWS00903675	Registration Number: 357332966
Candidate Score: 730	Pass/Fail: FAIL

The AWS Certified DevOps Engineer - Professional (DOP-C01) has a scaled score between 100 and 1,000. The minimum scaled score needed to pass the exam is 750.

We regret to inform you that you did not achieve the passing score required on the AWS Certified DevOps Engineer - Professional.

This is the score of my first attempt, it still hurts 💔

Don't limit by the examples I used to prepare. Think about new tasks you can solve, push buttons if you are not sure what they do, break things and fix them.

There are services which are used to confuse you and by knowing that these services exist and what they roughly do, you can guess the correct answer. There are not many of them but they are there to test your overall knowledge of AWS so when you need to use them to solve a problem it is good to know they exist.

every question because as you can see in the picture, even one question can change the result. At the end of the day, a FAIL is a fail, you can not say I almost passed the exam. **You can but it has no meaning** 🤦‍♂️🤦‍♂️

Here is the list of services you should be aware of and more or less what they do

1. **AWS Config**
2. AWS Managed Services
3. **AWS Step functions**
4. AWS Service Catalogue
5. **AWS Trusted Advisor**
6. AWS Macie
7. AWS GuardDuty
8. **AWS Inspector**
9. AWS CloudSearch (referred to as CS in the exam)
10. AWS Server Migration Service (referred to as SMS in the exam)
11. AWS DirectConnect
12. **AWS Organisations**
13. AWS Quick Insight

Just go through them, read FAQs, understand what they are and what they do at a very high level. After you think you know all these services, take some paper and a pen and write down the opposite of each service what it is for and what it does. If you don't do it you are going to regret it during the exam. 🤭Give more attention to the ones that are in **bold.**

this re:Invent video to get a deeper understanding of how indexes work and this one to understand what DynamoDB consists of under the hood.

Conclusion

Although it is not rocket science it is really a difficult exam. It is just a ridiculous amount of stuff you should learn, know and be able to use appropriately. Don't rush it, give it time to digest the information you consume. Use all your available time, lunchtimes, learning times (if you have any at work), your evenings, weekends.

It took me a month to get 73% score from the date I decided I wanted to pass it, along with 2+ years AWS experience and another month to successfully pass it with 90% score. It is all about a matter of time and effort.

Most importantly, take it easy, failures are the pillars of success

AWS Certification DevOps Aws Certification Aws Devops