Welcome future AWS Cloud Engineer!

Cloud is no longer just a **buzzword**, it's the **future**. So congratulations on your interest in the extremely addictive topic "Public Cloud/AWS", gaining more and more importance every day.

That you are here means you are open to the new mindset required to master complex cloud related problems and we will give you a kickstart on the necessary basics you'll need for this.

If you already know about networking, clusters, HA and dual datacenters that's great – but don't start skipping stuff, AWS has its peculiarities and changes every day - you don't want to miss something important, right? If you're new to all of this, don't worry, we'll get there.

During this Program you'll learn a lot of new and interesting stuff.

This training is kept in English because most of the AWS documentations and communities are in English anyway and we are improving it together with other Claranet group members. Weekly discuss will still be in German language ;)

Each part is split into theoretical learning through docs or videos and some practical exercises/structured hands-on labs. At the end you will have to answer some questions.

We will discuss your answers during our mandatory "**AWS Sprechstunde", every Wednesday from 09:00am to 10:00am in London** (The Room, not the City 😉).
Please be prepared!

Before we start, please have a look at the following website:
https://www.expeditedssl.com/aws-in-plain-english

It lists most of the AWS services and translates them into something understandable.
You might want to keep this page open during the next days, it will help you a lot in the beginning!

Check that you have access to the **Claranet AWS Test Account**

Option 1 **(via Office AD):**
Open your browser
Go to https://awstest.de.clara.net
Login with your ActiveDirectory account (Office AD)
Select the **ClaranetAdministratorAccess** role in the **claranet-de-test** account.

If you don't see any roles you might not be a member of the AD Group **DE-AWS-TEST** - please open a ticket (Access to a tool: AWS Test) in our Inhouse Portal.

Option 2 **(via MGT-AD):**
Open your browser
Go to https://aws.de.clara.net/
Login with your MGT-AD account
Select the **ClaranetAdministratorAccess** role in the **claranet-de-test** account.

If you have other technical problems with AWS mail us: journey-to-the-cloud@de.clara.net

Feel free to start some instances there and play around with them, but **please don't forget to stop or terminate them afterwards** as we need to pay for all the costs that occur.
Please always use the smallest instance size possible for your purpose!
Do not register domains. Use new subdomains of claranet.io, available in the test account.

Please **tag your resources** with "Creator" and your username in the form of "john.doe".
Please note that untagged resources might be stopped or deleted without warning!
Feel free to use Resources which produces Costs, but don't forget to shut them down once not more needed!

Feel free to watch one of these short videos and do a LAB on that topic whenever you have the time: https://aws.amazon.com/training/intro_series/

Part 1: Computing

OK, let's go. Let's start your journey!

We'll begin by watching this short introduction to Amazon Elastic Compute Cloud (EC2):
https://www.youtube.com/watch?v=TsRBftzZsQo

What about some practical experience right now? Start your first lab!
LAB: Introduction to Amazon Elastic Compute Cloud

Finally, watch the Amazon EC2 Masterclass deep dive session:
https://www.youtube.com/watch?v=ujGx0tiI1L4

Labs

Let's do some more labs:

LAB: Creating Amazon EC2 Instances with Microsoft Windows

LAB: Bundling Amazon EBS-Backed AMIs

Lab IDs change all the time due to updates. If so, search for the name and let us know!

Questions

Can you answer the following questions?

1.1 Please explain why using the T2 instance type could be a problem?
1.2 Why could it still be a good choice?
1.3 What is the difference between T2 and T3 and where should these be used?
1.4 What is the difference between previous generations and the new T4g?

1.5 Could you explain the difference between HVM and PV Virtual machine?
1.6 Which technology should we use? Why?

1.7 Could you describe shortly what the Enhanced Networking feature in an EC2 context is?

1.8 Is it possible to change (detach/attach) the IAM role of an EC2 instance after creation?

1.9 What is a UserData script?
1.10 What is cloud-init?

1.11 How many inbound entries could you have in a security group?
1.12 How can the limit be modified?

If you start instances in the Claranet AWS Test Account, please don't forget to terminate them afterwards. Noticed something that's not working as expected? Feedback is always welcome! Mail us, you'll find the address in the introduction.

Part 2: Networking

The High Level Picture
Learn where and how data centers are built by AWS around the world:

Global infrastructure
https://aws.amazon.com/about-aws/global-infrastructure/

Region and availability zones
http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html

The region where your data and your application are stored could have
a significant impact on latency and moreover on the user experience.

You should choose the region carefully when building a new platform.

You should also notice that not all services are available in all AWS regions.

This link summarize what service you should expect on which region:
https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services/

Networking inside a region
Learn what a VPC is and why we need it:

Description of VPCs
https://aws.amazon.com/vpc/
https://aws.amazon.com/vpc/details/

Introduction to Amazon VPC
https://www.youtube.com/watch?v=Fp_vwYBBQ1s

AWS re:Invent 2015 | (NET201) VPC Fundamentals and Connectivity Options
https://youtu.be/5_bQ6Dgk6k8 (or try the VPC Deep dive: https://youtu.be/B8vnhRJDujw)

Private and public DNS: Introduction to Route53
https://www.youtube.com/watch?v=e2xLV7pCOLI

Labs
Do them in claranet-de-test Account!

LAB: Introduction to Amazon Virtual Private Cloud (VPC)

LAB: Introduction to Amazon Route 53

Create a VPC in the **Claranet AWS Test Account** (see the introduction on how to login).
Now add a private Route53 zone and attach it to it. How can you test the functionality?

Ready to answer some questions? Part 2 continues on the next page.

Part 2: Networking

Questions

2. Explain what the difference between a public subnet and a private subnet is…
2.1 …from a functional point of view?
2.2 …from a technical point of view?

2.3 What setting can turn a private subnet to a public one?

2.4 Describe the old and the new approach to build a NAT Gateway service for your VPC.

2.5 What is the cost difference between both approaches? Calculate with T2.small instances for the old approach.

2.6 Please classify Security Groups, NACL and Linux iptables filtering mechanism from the first applied security layer to the latest.

2.7 When would you use which? Why?

If you start instances in the Claranet AWS Test Account, please don't forget to terminate them afterwards. Noticed something that's not working as expected? Feedback is always welcome! Mail us, you'll find the address in the introduction.

Part 3: Security

Watch the following short video on IAM users, groups and roles:

Introduction to AWS Identity and Access Management (IAM)
https://www.youtube.com/watch?v=Ul6FW4UANGc

AssumeRole or SwitchRole is a very powerful feature in multi account setups. Read about it:
http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use.html

Have a quick look on this very detailed Whitepaper about Security on AWS:
http://d0.awsstatic.com/whitepapers/Security/AWS_Security_Whitepaper.pdf

Reading it all would be a bit too much for now, but it's important to know that such a thing exists and where to find it in case you need to look something up.

Do you know CloudFront? It's Amazons CDN:
https://www.youtube.com/watch?v=YHNhWd8JAy0

Labs

LAB: Introduction to Amazon CloudFront

LAB: Using Open Data with Amazon S3

Try to access our customer test account **testkunde-testprojekt-test (400999792231)** through the AWS Console using "switch role" starting from the **claranet-de-test account.** Try to assume the role **ClaranetReadOnlyAccess.** Once you changed accounts, click around a bit and then go back. How do you do this using the without logging out? Notice that successfully switched roles remain in your history (as long as you don't clear your browsers cookie cache).

Now let's do this the Claranet Germany way: Go back to https://aws.de.clara.net and change your account and role here. Go to https://aws.de.clara.net again and switch back to our test account as an Administrator. Which way is easier? Which is faster? Which do you prefer?

Create a **Cloudfront distribution** and a **S3 bucket** in the **Claranet AWS Test Account. Attention: Use the standard region for Cloudfront and S3 to speed things up! (Why?)**

We want to limit access to S3 to the CloudFront distribution only. Store a simple image (the Claranet Logo for example, see below) in a new S3 bucket and then answers the questions.

Claranet logo: https://www.claranet.de/sites/all/themes/claranet_responsive_2015/logo.png

Ready to answer some questions? Part 3 continues on the next page.

Part 3: Security

Questions

3.1 Provide the link to get access to your image through CloudFront

3.2 Provide the link to the same image on S3 which is not directly accessible from the Internet (only with the correct Access/Secret keys)

3.3 Please provide your policy from 3.1 and 3.2.

3.4 Please write a policy which would give access to a specific S3 bucket for a specific IAM user/role.

If you start instances in the Claranet AWS Test Account, please don't forget to terminate them afterwards. Noticed something that's not working as expected? Feedback is always welcome! Mail us, you'll find the address in the introduction.

Part 4: Storage

There are basically two types of storage in AWS:
Block level storage (EBS) and File/Object level storage (S3)

Let's start with Block level storage by watching this short introduction:

Introduction to Amazon Elastic Block Store (EBS)
https://www.youtube.com/watch?v=77qLAl-lRpo
and
https://www.youtube.com/watch?v=S0gzrxsVQHo

Let's have a look at the details (and don't forget the pricing!):
https://aws.amazon.com/ebs/
https://aws.amazon.com/ebs/details/
https://aws.amazon.com/ebs/pricing/

For real technical insights now watch this deep dive video on YouTube:

Amazon EBS deep dive
https://www.youtube.com/watch?v=BuJa6Vl8cn8

And in comparison File/Object level storage:

Introduction to Amazon S3
https://www.youtube.com/watch?v=77lMCiiMilo

Simple Storage Service (S3)
https://aws.amazon.com/s3/
http://aws.amazon.com/s3/details/
https://aws.amazon.com/s3/pricing/

Now watch these videos about the S3 object store:

Amazon S3 Masterclass
https://www.youtube.com/watch?v=VC0k-noNwOU

Amazon S3 Deep dive and Best Practices
https://www.youtube.com/watch?v=rHeTn9pHNKo

EBS vs. EFS vs. S3
https://www.youtube.com/watch?v=m9mKzdBE90I

Labs

LAB: Introduction to Amazon Simple Storage Service (S3)

LAB: Introduction to Amazon Elastic Block Store (EBS)

LAB: Working with Amazon Elastic Block Store (EBS)

OPTIONAL:
Try to setup a CloudFront Destribution on AWS with your Content.
Ready to answer some questions? Part 4 continues on the next page.

Part 4: Storage

Questions

4.1 In which cases would you use S3 instead of EBS?

4.2 Your customer needs a 1TB Volume of EBS storage with 3000 IOPS. Which EBS technology would you use? Why?

4.3 Your customer wants you to install an EC2 instance in a private subnet. Your EC2 instance needs to reach a S3 bucket to store a dump of its data. Describe two architecture designs that answer this problem. Explain which one is the most efficient.

If you start instances in the Claranet AWS Test Account, please don't forget to terminate them afterwards. Noticed something that's not working as expected? Feedback is always welcome! Mail us, you'll find the address in the introduction.

Part 5: RDS Databases

Of course you can always run your database on an EC2 instance,
but AWS also offers several database types as a managed service.

You should try to use these whenever possible.

Let's start by watching this short introduction:

Introduction to Amazon Relational Database Service (RDS)
https://www.youtube.com/watch?v=eMzCI7S1P9M

Now, let's have a look at the docs:

https://aws.amazon.com/rds/
https://aws.amazon.com/rds/details/
https://aws.amazon.com/rds/pricing/

We'll continue with some more details:

Amazon RDS for MySQL: Best Practices
https://www.youtube.com/watch?v=eHg8LD5KNC0
Optional:
https://www.youtube.com/watch?v=-my0q7aADcY

Amazon Aurora: Amazon's New Relational Database Engine
https://www.youtube.com/watch?v=g5N_kFRqCh8

And finally… there's a deep dive video, of course:

Deep Dive on Amazon Aurora
https://www.youtube.com/watch?v=TJxC-B9Q9tQ

Labs

LAB: Introduction to Amazon Relational Database Service (RDS)

OPTIONAL:
Setup an EC2-Instance with MySQL
What are the Pros and Cons (RDS vs. MySQL on EC2)

Ready to answer some questions? Part 5 continues on the next page.

Part 5: RDS Databases

Questions

5.1 There is a complete availability zone outage in the AZ where your master DB server is running. Will there be an outage? If so, how long will it last?

5.2 You are migrating an on-premise application into AWS. The application uses one read/write DB and one read-only replica. The app is very read heavy, and the customer would like to retain this pattern. Assuming r3.xlarge instances for both r/w and ro nodes, how much would this cost to run on MySQL RDS? Would it cost the same on Aurora? If not, why not?

5.3 Your customer has an on-premise infrastructure that he would like to migrate to AWS. This infrastructure has a MySQL (5.6) database with 500 GB of data. He wants to minimize the time of the switch between his old infrastructure and the newest one in AWS. Could you describe the best scenario in order to do that?

If you start instances in the Claranet AWS Test Account, please don't forget to terminate them afterwards. Noticed something that's not working as expected? Feedback is always welcome! Mail us, you'll find the address in the introduction.

Part 6: Load balancing and Auto Scaling

At this time you should refresh your memory – do you remember the OSI model?
Have a look at layers 3, 4, and 7 – what's happening on each level?

Load balancing

There are many ways to load balance traffic across a farm of servers. You could spawn a HAproxy on top of an EC2 instance, use Nginx or even a F5 BigIP available on the market place. Or… you could just use Amazons Elastic Load Balancing Service! It's called ELB:

https://aws.amazon.com/elasticloadbalancing/
https://aws.amazon.com/elasticloadbalancing/classicloadbalancer/pricing/

Next to this "Classic Load Balancer" there's now a new star on AWS:

https://aws.amazon.com/elasticloadbalancing/applicationloadbalancer/
https://aws.amazon.com/elasticloadbalancing/applicationloadbalancer/pricing/

You should also watch the following deep dive from AWS re:invent on this topic:

AWS re:Invent 2015 | (CMP401) Elastic Load Balancing Deep Dive and Best Practices
https://www.youtube.com/watch?v=VIgAT7vjol8

Auto Scaling

Auto Scaling was the original attention grabbing feature of AWS. For many on premise customers, this is still a key factor for choosing to move into public cloud.

We should use Auto Scaling everywhere, even when we only want a single instance:

https://aws.amazon.com/autoscaling/
https://aws.amazon.com/autoscaling/details/
https://aws.amazon.com/autoscaling/pricing/ *Surprise ahead! ;)*

Watch this video for more information:

All You Need To Know About Auto Scaling
https://www.youtube.com/watch?v=4trGuelatMI

Labs

LAB: Introduction to Elastic Load Balancing

LAB: Working with Elastic Load Balancing

LAB: Introduction to Amazon EC2 Auto Scaling

Ready to answer some questions? Part 6 continues on the next page.

Part 6: Load balancing and Auto Scaling

Questions (some more as usual as a solid understanding of this topic is *really* key)

6.0 (Most important question ever) Why should we use Auto Scaling everywhere? Why would we use it even for single instances of which not more than one will ever exist for sure?

6.1 Imagine you need to update all web servers to a new AMI. What is the process for doing this, assuming web servers are created from an Auto Scaling group?

6.2 A customer has a PHP app that is very memory heavy, but doesn't use much CPU. How can we implement Auto Scaling? Why do we need to this? What's the problem for AWS?

6.3 How could an Auto Scaling setup for his PHP based webapp look like given that he wants to release a new version of his application

- 6.3.1 only once in the beginning of the project and maybe once or twice a year
- 6.3.2 several times a month (let's say up to 5)
- 6.3.3 up to 10 times a day

6.4 What would change if the application includes user uploaded and generated content?

6.5 When using Auto Scaling, what would you put in the AMI and what would you script?

6.6 Why could using t2 instances in an Auto Scaling group triggered by CPU usage become a problem under heavy, constant CPU load?

6.7 What do we need to do under all circumstances regarding AWS provided AMIs?

6.8 The PHP based Auto Scaling webapp needs a database. Can the database be part of this Auto Scaling group? Why?

6.9 The database fails. You configured an ELB health check for the Auto Scaling group earlier on the front page (index.php). This URL now returns the HTTP error 501, saying "DB timeout!" because of the database failure. What happens to your Auto Scaling group?

6.10 An application update involves a change of the database scheme. Assuming the database is not part of the webserver Auto Scaling group where we are going to deploy the new version of the application using rolling upgrade, how could this become a problem for the application anyway? How could we solve this?

6.11 What will probably happen during the update as well if you take all information from the other questions into account? What could you do to prevent this from happening?

6.12 What are the differences between the Classic Load Balancer and the new Application Load balancer from a technical point of view? What features does it offer additionally?

6.13 From an economic perspective (regarding costs): How is the pricing model different? When should we use which one?

If you start instances in the Claranet AWS Test Account, please don't forget to terminate them afterwards. Noticed something that's not working as expected? Feedback is always welcome! Mail us, you'll find the address in the introduction.

Part 7: Infrastructure as code

Here at Claranet we believe that infrastructure should be managed the same way as code is. Using a language that lets you describe the infrastructure and a tool to populate it opens a window to a whole new world. It allows us to:

- Have a complete change history for everything in the infrastructure, including a commit message which hopefully explains why a change was made
- Create multiple copies of an environment with consistency between each, and a method for testing upgrades (dev->stage->production)
- Create re-usable modules we can deploy to multiple customers, allowing us to complete work much more quickly than a customer could complete it themselves and with less tedium for the SREs involved

AWS provides CloudFormation, and some customers use it to build their environments. We decided to build environments with Terraform. Find out why: https://www.terraform.io/

Terraform

Watch the following introduction:

Seth Vargo on Hashicorp Terraform
https://www.youtube.com/watch?v=pKIFHtO2Y7I

And here is some more in-depth theory:

Applying Graph Theory to Infrastructure as Code
https://www.youtube.com/watch?v=Ce3RNfRbdZ0

Labs

Use the Claranet AWS Test Account to create a t3.nano instance and install Terraform on it:
https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli

This is now your management host. Create another t3.nano instance using Terraform:
https://developer.hashicorp.com/terraform/tutorials/aws-get-started/aws-build

Change this instance by editing the Terraform code:
https://developer.hashicorp.com/terraform/tutorials/aws-get-started/aws-change

Destroy your second instance using terraform:
https://developer.hashicorp.com/terraform/tutorials/aws-get-started/aws-destroy

Congratulations, you successfully built, changed and destroyed your first environment with Terraform! Now please delete your management host to avoid further costs. Thank you!

Ready to answer two short questions?

Questions

7.1 Explain the importance of the Terraform state file.

7.2 Think of at least two ways to work on an environment built with Terraform together with a colleague. How can you share the state file?