

Hi, these are the notes I took from the lectures. I think this can be very useful to refresh your memory before the exam. I wrote them very fast and there may be typos, this will not make sense to read without watching the lectures, so read this after. Let me know if you find any mistakes.

PART 1: Classic Managed AWS

IAM

You need to select a region for all services besides IAM and S3

Root user is created with the account and should not be used.

IAM: User, Groups(teams, users) and Roles(machines). You apply permissions using policies in JSON to users, groups and roles. A policy has resources events and actions.

IAM Federation is using SAML standard to use your IAM solution for example LDAP active directory.

To log into EC2 instance use `ec2-user@publicIP` use `-i` and then pem file with the key, run `chmod 0400` to give the right permissions to the pem file.

Use tag with key Name so you can add a name so it is show in the console.

Security groups are logical groups or firewall rules for the EC2 instances. Many to Many relation. Locked down to region. By default all inbound traffic is not allowed and all outbound allowed.

You can reference other security groups to extend functionality.

A elastic IP is a public IP that you own and never changes, you can use it to mask failure or if you need a constant IP but is better to setup DNS names and use random IPs.

Use load balancer instead of setting IPs.

You get bill for elastic IPs.

Use EC2 user data field to execute programs during provisioning, so you can add your apps to run at bootstrap like installing Apache. Is used to bootstrap the instance and only runs once. Runs with root user. Click Advanced details and paste the script you want.

AMI are EC2 OS images, you have Amazon on linux based on Fedora, Red Hat, windows... you can create your own images and then create all your instances with them. You need this if you have proprietary software, extra security or federation is required to access your company LDAP.

Customs AMI are made for a specific region.

Instance names start with letters to give meaning: R -> RAM, G -> GPU, etc... M are the balanced ones and T the burstable, you have burst credits you can get in advance and if you get a spike, your CPU will burst and handle them until you run out. Credits are given as part of the instance and once used you recover them over time, you will never pay unless you do T2 unlimited. There is T2 unlimited which if you run out of credits then you will be charged.

Ec2 are billed per second.

ELB: spread load, expose DNS handle failure, SSL termination, cookie stickiness, HA zones and separate public from private traffic.

AWS guarantees HA on the ELB and maintenance.

3 ELB: classic which is not recommended and then you have the v2 version which are application and network load balancers. You can setup private or public ELB.

Application load balancers ALB are level 7 and they can handle multiple http apps even in the same machine, you can use the url or hostnames to balance. It is great for microservices, it support dynamic port mapping to run multiple containers on the same EC2 instance. The new app balancer is much cheaper because it can handle multiple apps.

You can bundle multiple ec2 instances into target groups that are the same app/route. So you can scale and maintain cookie stickiness.

You can enable stickiness at the app level transparent to the app.

ALB support HTTP/s and websockcets, classic load balancer does not support web sockets.

The application servers doesn't see the IP of the client directly, it is in the X-Forwarded-For header set by the ELB. This is because the ELB does connection termination including SSL then redirects so the EC2 see the ELB ip, to get the client use the header.

Network load balancers CLB are level 4, lower level and are for TCP traffic. Really high performance and low latency.

CLB/ALB both have ssl termination and healthchecks.

ALB is great for docker because it can route based on port and path

All ELBs have static host names, but not IPs, do not use IPs.

ELBs can scale but not instantaneously. NLBs see directly see the client IP, ALB use forward headers. 503 error means ELB is running out of capacity.

You can use auto scaling to automatically register new instances to the load balancer. Scale out is adding instances and scale in removing. You need to define auto scaling groups **ASG** and the balancer will use it to register new instances.

ASG needs the same type of info as EC2 including AMI, EC2 user data, EBS volume, security groups, etc. You also need to specify min, desired and max capacity. You also need to specify the network and subnet settings, the load balancer info. and the scaling policies, that is when to scale in or out. This can be done using Cloudwatch which is the monitoring service in AWS, they do have alarms. so you can trigger scale in/out based on the triggers in the alarms. Some of the alarms can be based on avg cpu

usage, number of request in the ELB, network usage, etc. You can also create custom metrics in cloud watch. Even you can scale based on a schedule. IAM roles attached to an ASG will be assigned to the EC2 instances. ASG is free, you pay for the instances. If you create ASG for your instances, with a min number if you terminate them, the ASG will provision a new one to meet the min req. ASG will terminate unhealthy instances.

Load balancers and ASG are under the menu in EC2. First you need to create a launch template and then the ASG. The template is to create new EC2 instances, so the config is the same as for creating an EC2 instance, this way you enforce that all of them are the same type. When creating a ASG it is recommended that you go to advanced details and select load balancing so it works with the ELB, just select the target group previously created for the ELB. You can reuse the health check for ELB. You can attach and detach instances to the ASG, it is best practice that all ec2 instances are part of an ASG and ASG are part of target group for the ELB. So load balancers and ASG work together.

EBS is elastic block storage which is a network drive that you can attach to instances while they run, so you keep the data if the instance is terminated or if you want to store data and persist data. You can attach the EBS volume and then attach it to a different volume so you keep the same data between instances. EBSs are locked to availability zones. You cannot transfer it between the data centers, you can move it by taking a snapshot and then copy the data to another AZ. EBS have a provision capacity(size and IOPS). You can attach multiple volumes to the same instance. You have EBS types such SSD, HDD and the price differs. GP2 are the avg which are cheap and SSDs. You can increase EBS capacity at any time. You can take snapshots to backup data or to transfer between AZ. You can schedule snapshots. You can create an EBS encryption, all snapshots are encrypted. The data between EBS and the instance is transferred encrypted. All volumes created from snapshots are encrypted. Data is encrypted at rest and in flight, it has slow latency. Copying an unencrypted snapshots forces encryption automatically. Some EC2 instances do not have EBS, they use instance store, which means that the root volume is attached to physical drive, not network like EBS, you get better performance but you lose the data on termination and you cannot resize, also backups must be handled by the user. EBS backed are recommended unless you really need the performance. Each EBS volume can be attached to just one instance and one instance can have many volumes. EBS are locked to the AZ, you can migrate by taking a snapshot and copying it. Snapshots use IO and impact performance, take snapshots when the load is low. Although EBS volumes are persisted and you can attach/detach them from instances, by default the EBS volume will be terminated with the instance, you need to click the check box during creation to tell AWS not to terminate the volume.

Route 53 is a managed DNS which is a collection of rules and records to help clients reach servers using urls. the most common records are A: url to IPv4, AAAA URL to IPv6, CNAME for url to url and Alias url to aws resource. Route 53 can work with public or private domains. It also has load balancing using DNS not IP, called client side load balancing, health checks and routing policies based on geolocation, failover, latency, etc. Use Alias instead of CNAME because it is faster.

RDS is a relational database managed service, it is a managed relation database, it is a service that manages an underlying RDBMS, not a DB itself like DynamoDB. You can use Postgres, Oracle, MySQL, MariaDB, SQL Server and Aurora which is AWS proprietary. Because the DBs are managed by RDS you get lots of things for free like patching, backups, monitoring, DR, scaling, maintenance, replicas, etc. but you cannot access directly using ssh because they are AWS managed.

You can create up to 5 read replicas for RDS across any AZ or region. Replication is async so there could be a small delay, eventual consistency. Replicas can be promoted as its own DB so it can write. Applications must update the connection string to read from replicas, so it is not totally transparent.

Multi AZ is used for DR, it does sync replication between the master and stand by node which is not live, it takes over just if the master fails. It uses DNS failover, so both DBs have the same dns name, this is done transparent by AWS. So AZ replicas are read only live replicas for performance and Multi AZ is standby for HA. RDS uses nightly backups by default with 7 days retention policy, you can restore at any point in time. It also captures logs in real time. You can also do manual snapshot and they will not be deleted. RDS uses encryption at rest using AWS KMS AES 256, you can use SSL for in flight encryption. To enforce SSL it differs on the RDBMS. In Postgres rds.force_ssl=1 to connect to an SSL DB you need to enter the SSL trust certificate that can be downloaded from AWS. You should deploy RDS DBs in a private subnet. They also use security groups like EC2 to enforce who can access. IAM policies are used to control who can manage the DBs. MySQL and Aurora can leverage IAM users instead of having a separate username and password.

Aurora is not a full DB, it is based on MySQL and Postgres, so drivers are compatible it is cloud optimized performs up to 5x better and can dynamically grow. Use Aurora if you don't care much about portability. It can have up to 15 replicas and has instant failover but it is a bit more expensive. Without Aurora you need to setup the replicas and failover.

ElasticCache is a managed cache service using Redis or Memcached to improve read performance or add state to stateless apps such as serverless. They use sharding, multiple replicas, multi az failover, etc. AWS takes care of all maintenance: monitoring, patching, failover, etc. Elastic Cache is used a lot with lambda or stateless micro services to use it for session management or JWT token management. Redis is the most popular choice, it is in memory but persistent into disk (async) so it survives restart. Great for user sessions, distributed states, it has pub/sub queue. It has multi az failover. Memcached is 100% in memory, but Redis is more popular.

A **VPC** is your own private cloud, you can create them inside a region. This isolates you from the rest. Each VPC contains subnets, each subnets is inside an AZ. You usually have private subnets to have the DBs and services and the public ones to have the public HTTP servers. It is common to have several subnets inside each AZ. In public subnets you put: Load balancers, api gateways, files and static content, and authentication. In private you have the servers and databases. Public and private can talk to each other. Public have access to the outside, private don't. You can use a VPN to access your VPC and see all the private network. VPC flow logs are used to monitor the inbound and out bound traffic from your VPC. You can have several VPC in your account, one per region. You can peer VPCs so they look like they are part of the same network even if they are in different regions.

S3 is the simple storage solution of AWS and the most famous service used for small and big data files. Buckets are object containers that can be folders and files. Buckets must have global unique names although their are defined at a region level. Names must be underscore and not an ip. Objects have keys which is the full file. Internally there are only objects inside buckets, but the UI offers folders but these are just logical paths which are the keys. The max value of an object is 5TB. More than 5GB

you need to use multi part upload. You have meta data and tags, there is also a version ID because you get versioning, but you need to enable it.

S3 is a global service that's why the bucket ID is unique, you do not have a region, however when you create a bucket you need to select a region. This is because the bucket will have a unique DNS name and these are global.

File versioning is enabled at the bucket level for all files. If you update a file a new version is created automatically. You can enable versioning at any time, if there are files already their version will be set to null and any changes from there will have a version.

4 encryption methods on s3: SSE-S3 uses keys managed by AWS, SSE-KMS uses AWS KMS service to manage keys, SSE-C you provide your own keys for encrypt, Client side encryption you manage.

SSE-S3: managed by AWS, AES-256, encrypted servers side and must set header x-amz-server-side-encryption: AES256 when you send file to S3, Amazon provides the key.

SSE-KMS: uses KMS service so you have more control on the expiry and audit trail you also need to pass the header. You manage the lifecycle.

SSE-C: server side encryption using your keys that you manage, AWS doesn't store it, so you need to pass the public key in the upload request using headers, because of this you must use HTTPS.

Client side encryption: you encrypt and decrypt AWS doesn't do anything.

AWS exposes https for encrypted data and http for non encrypted. Encryption in flight is called SSL/TLS.

Two types of S3 security user based (IAM policies, based on API calls), Resource Based: bucket policies which is the most used and are wide rules you set in the console. Object ACL finer grain and Bucket ACL. Bucket policies are json files with resources, actions and effect(allow/deny) and principal (account or user). you can grant public access, force objects to be encrypted on upload and grant access to other AWS accounts. S3 supports VPC end points for instances without public access so instances in the private network can access the file. You can have S3 access logs and store them in other buckets. API calls can be logged in cloud trail. MFA can be used for extra security. You can create signed urls which are available for a limited time only. There is a policy generator that you can use to generate JSON policy documents: <https://awspolicygen.s3.amazonaws.com/policygen.html>

You can setup websites in S3 using s3-website prefix, make bucket public.

If you request data from another bucket you need to enable cors. Is common to have images in another bucket but you don't want public access but thru website bucket so enable cors to set origin.

S3 uses eventual consistency you can put and read and get an object immediately but if you do a get first and got 404 you may get it right after you put it, same as update or delete.

For the best S3 performance you want the highest partition distribution. Historically it was recommended to have random characters in front of the key name like "g65d_my_folder/" so the data is partitioned properly. Never use dates as keys since the partition will be very similar. Since July 2018 you don't need random characters and the performance is much better > 3500TBS but this is not in the exam.

More than 5GB you need multipart uploaded to maximize bandwidth and decrease time on retries. CloudFront is used to cache S3 objects. If you are in a different region and need to do a big file upload use S3 transfer acceleration which uses edge location to upload temporarily to a local endpoint improving performance. If you use KMS encryption you may see a small performance decrease because KMS has bandwidth limits.

CLI: You install the CLI using python, then use aws command. First create access keys in IAM and use aws configure to enter credentials. Then start using aws, for example aws s3 ls.

You can use the cli inside an EC2 instance but this is a bad practice, do not use your credentials anywhere. To use the cli in the EC2 use IAM roles which can be attached to EC2 instances, roles come with a policy authorizing what the ec2 can do. You can assign a cli role to ec2 instance to use cli. aws cli is already installed in aws linux amis but do not use aws configure to set the keys, you can set the region if you want. You need to create a role and attach it to the ec2 service, once you have the role click on the instance and select attach role.

You can create your own policies and assign them to roles, you can also create inline policies specific for just one role, but this is not recommended. Policies have versions to track changes.

There is a AWS policy simulator to test your policies. Many commands have the --dry-run option to test calls without being charged, this is perfect to test for permissions.

The STS command line tools is used to decode errors: aws sts decode-authorization-message. You need DecodeAuthorizationMessage permission in the role to run the command.

EC2 instance metadata is a powerful tool so instances learn about themselves and what permissions they need without using an IAM role for that purpose. The metadata is the info about the instance and can be retrieved at <http://169.254.169.254/latest/meta-data> which is a private IP. This metadata is heavily used for automation and to run scripts.

AWS has a SDK to access its services. Java, Python, Node.js, C++, etc. AWS cli is based on the python sdk. The SDK is used to deal with DynamoDB and other services. To use the SDK use the default credential provider chain in the user directory aws/credentials. You can also use ENV variables not recommended but useful for docker images. Never use credentials in the code. For API calls from the SDK AWS implements exponential backoff so if you make a request and it fails the SDK will automatically retry but each time it waits double to not overloaded.

If you have multiple AWS accounts you can use profiles using aws configure --profile, so you can use the CLI with multiple accounts.

Elastic Beanstalk is a developer centric simple view to manage your apps which leverages on the existing components like elb, rds, AGS, etc which are easier to manage, it is a CI tool. Beanstalk manages your instances like os, configuration, etc and also runs the deployment strategy defined by you. It is key for CI. It has 3 components: your app, the app version and the environment (test, qa, prod). You can promote a build through the environments or rollback. It is free to use. Many platforms are supported including docker. Each app can have many envs, each env many versions. You can choose web server or worker type (batch). Beanstalk can create a DB but it is better to externalize it since if you delete beanstalk you lose the DB.

Deployment Modes:

- Single Instance: great for dev, DNS name is Elastic IP
- HA with load balancer: ELB with the DNS name and many instances. Multi AZs. Auto scaling group.

4 kind of deployments for HA:

- All at once: very fast and simple but small downtime. No additional cost
- Rolling: You update some of instances call bucket and once is healthy move to the next. You are under capacity during the update but no additional cost. It can be a long deployment.
- Rolling with additional batches: like the other but spins new instances to maintain same capacity during the update and then once completed the old ones get deleted. Long deployment and extra cost. You have a new extra bucket, so it cost extra.
- Immutable: Spins new instances in a new ASG and when done it switch the ASG. zero downtime and brand new instances, high cost because you replicate cluster and it takes long but you get a quick rollback, you just switch to the old ASG.
- Blue/Green: create a brand new environment, test it independently. Then use Route 53 to start directing traffic in bits until you are happy.

You can use EB extensions files to configure the deployment. The directory must in the root of the src code and must be called .ebextensions and the config must be in yaml or json. the extension must be .config like logging.config. You can use these files to modify default setting or to add resources like RDS or DynamoDb.

There is also a EB cli specify for Elastic Bean and you can eb create, status, [etc.to](#) manage environments and versions. Very helpful for CI/CD. Eb relies on CloudFormation which is a service. Usually you resolve dependencies when you install like npm install, but when you have many instances, and you need to do npm install in all of them, it is better to do npm install in one machine and then create the zip file with the dependencies inside.

AWS CI equivalents: Github->CodeCommit, Jenkins->CodeBuild. To deploy you can use BeanStalk or CodeDeploy. AWS CodePipeline is used to orchestrate all the tools CodeCommit, CodeBuild, BeanStalk etc.

CodeCommit is a private GIT with no size limit and fully managed and secure. You can integrate it with Jenkins. Security: SSH keys, HTTPS (generate credentials), it supports MFA. It uses IAM policies for authorization. Repositories are encrypted at rest using KMS. In transit uses HTTPS. For cross account access do not use ssh keys or aws credentials, use IAM role and then use AWS STS to access. Both github and code commit integrate with CodeBuild. **CodeBuild** integrates with IAM roles with github you have other users. Github Enterprise is hosted in your servers. CodeCommit UI it is worse. CodeCommit can integrate with AWS SNS for notifications or AWS Lambda or CloudWatch event rules. Alerts for deleting, pushes, you can run AWS lambda functions to respond to events. CloudWatch rules are uses for pull request or when a comment is added. CloudWatch rules uses SNS under the hood.

CodePipeline: CD with visual workflow. Source: Github/CodeCommit, build, load test, deploy (AWS CodeDeploy, BeanStalk). It is made of stages each stage can have sequential or parallel actions. Stages: Build/Test/Deploy/LoadTest... You can define manual approval step. CodePipeline works with Artifacts. Each stage can create several artifacts which are passed and stored on S3 and passed to the next stage. Each stage changes will generate a CloudWatch event which can trigger sns notifications. CloudTrail can be used to audit AWS API call. You attach roles to pipelines and must have the right permissions to perform the actions.

CodeBuild: build and test like Jenkins. It is fully managed by AWS which handles provisioning and scaling. You pay for the usage only. It uses Docker under the hood and you can create your own images. It is secure, integration with KMS/IAM/VPC... it uses cloud train for audit.

Takes code from CodeCommit/github. Build instructions are defined in the code in the buildspec.yml file. Outputs log are output to S3 and cloudwatch logs. There are also metrics to monitor and get statistics. You can use cloud watch events and AWS lambda for alerts and notifications that can use SNS. You can reproduce CodeBuild locally to troubleshoot errors. Pipelines can be defined with CodePipeline or CodeBuild itself. CodeBuild support Java, Ruby, Python, Go, Node, .Net, etc.... You can use Docker to use any other platform. You have Source code with the buildspec.yml. CodeBuild will create a image with the code and run it on a container. There is an optional S3 bucket for cache. Artifacts are stored in S3. Logs are saved in S3 and CloudWatch. buildspec.yml must be in the root of the code. You can define env Variables, you can use secrets using SSM parameter store. You define phases: install(dependencies), pre build, build and post build. Artifacts are generated and stored in S3, cache is also based on S3. You can run CodeBuild locally using CodeBuild Agent.

CodeDeploy: deploy app into many EC2 instances which are not managed by Elastic Beanstalk. Similar to Ansible, Chef, Puppet... Each ec2 machine must run the CodeDeploy agent. The agent is continuously polling AWS codeDeploy for work to do. CodeDeploy sends appspec.yml which has the instructions with the source code, then the app is pulled from GitHub or S3, EC2 will run the deployment based on the yaml instructions and the agent in the EC2 instance will report success/error. EC2 instances are grouped by deployment group (dev,test,qa...). CodeDeploy can be chained into CodePipeline and use artifacts from there or other tools like auto scaling groups. CodeDeploy is more complex than BeanStalk but more flexible and supports blue green deployments but only on cloud, It supports lambda. It doesn't provision EC2 instances so they need to be ready. Deployment configuration: deployment rules for success or failures for example the minimum number of healthy instances. Compute platform: Ec2 or lambda. Deployment type: in place or blue/green. IAM instance profile needed to give EC2 the permissions to pull from S3/Github. Application Revision: application code + appspec.yml. Service role: Role for CodeDeploy to perform operations. Target revision: target deployment app version.

AppSpec file: File section: how to source and copy from S3/Github. Hooks: set of instructions to deploy a new version. The order is StopApplication, DownloadBundle, BeforeInstall, AfterInstall, ApplicationStart, ValidateService(it is healthy)

Configs for Code Deploy: one at a time, half at a time, all at once (quick but downtime), custom.

Failures: instances stay in failed state, new deployments will first be deployed to failed instances. To rollback redeploy old deployments or enable automated rollback.

Deployment Targets: Set of EC2 instances with tags, directly to ASG, mix ASG/tags or customization scripts using ENV variables. BeanStalk uses CodeDeploy under the hood.

For blue/green deployment there are BeforeAllowTraffic and AfterAllowTraffic hooks to test it is healthy.

CodeBuild-> buidspec.yml CodeDeploy-> appspec.yml

AWS OpsWorks is a configuration management service based on Chef that allows managing and deploying complex application stacks on AWS.

Cloud Formation: Infrastructure as Code. Traditional CI/CD is hard to reproduce in another region or different AWS account. Infrastructure as code is turning infrastructure into code. Automate infrastructure. Cloud Formation is a declarative way to provision infrastructure. We just tell them what we need (Load balancer, security groups, instances, etc) and cloud formation goes and does it for you. No resources are manually created, everything is automated and the declarative code can be in git. Changes in infrastructure are code reviewed. No additional costs just what you provision, you can estimate the cost with templates. You can have your whole environment provision and deleted, companies they do provision the dev envs at 8 and destroy them at 5pm to save costs. Separation of concerns: there are different stacks: vpc, network, and app stacks. There are lots of cloud formation templates with stacks you can reuse. You upload the templates in S3 and cloud formation will read it. Stacks are identified by names and if you delete the stack you delete everything (not good idea to provision your DBs there). There is a cloud formation designer but you can use yaml file with templates for automation using AWS cli. Building blocks: Resources, parameters, mappings, outputs, conditionals and metadata. You can have template helpers with functions.

Resources are mandatory and there are over 200. They can reference each other and they represent AWS components. AWS::aws-product-name::data-type-name. Everything you click in the UI manually is translated into the yaml format this way it can be automated. You cannot create things dynamically in Cloud Formation, it needs to be declared in the template.. There are few AWS services not supported but you can use AWS Lambda custom resources to use them. Parameters are used to pass inputs to the template, very useful to reuse templates. So they are like custom fields. Use Fn::Ref function to reference parameters like vpclId: !Ref MyVPC. !Ref can be used to reference other parameters or even reference resources. Pseudo parameters are custom parameters provider by AWS like AWS::AccountId.

Mappings are hardcoded values in the template, great to define env specific values. If you know the values in advance by deriving them from other values like region you can set up safety templates than using parameters. Use !FindInMap function to get values based on keys. Outputs are optional and can be used as inputs to other templates. You need to export them and then you can use them in other templates, console or cli. Each team can own a template like the network team can have the network template and export the vpc id as output for devs to refer. You cannot delete a stack if its outputs are used by other stacks. Use Export: block to export the value using a name, then use !ImportValue using the export name. Conditions: have conditions using equals and other operators. CreateProdResources: !Equals [!Ref EnvType, prod] ->if prod. So you define conditions and then use Condition: to use it. Intrinsic Functions: !Ref-> reference parameters. !GetAtt get attributes from resources, like Ref but ref only returns the ID of the resource but getAtt you can get anything. ImportValue-> previously exported. !Join -> join values to create resources. !Sub -> substitute values in Strings. By default is

something fails the whole stack gets deleted. There is an options to disable this. If you update and it fails it goes back to previous state. You can check the log to see the messages.

CloudWatch: Metrics, logs , events, alarms. Can send notifications using SNS.

AWS X-Ray: Similar to Istio, has distribute tracing, performance monitor, troubleshooting.

CloudTrail: Internal monitoring and audit changes of resources.

Cloudwatch has **metrics** for all services. Metrics belong to namespaces. Dimension is an attribute of a metric, we can have up to 10 per metric. Metrics have a timestamp. We can build dashboard. EC2 comes monitor with metrics every 5 min, if you want faster enable detailed monitor and pay for it. Memory usage is not pushed by default, you need a custom metrics. By default custom metrics are push every one minute. You can enable high resolution to go up to 1s. To send a metric you need to use the API call PutMetricData which uses the exponential back off strategy to handle error throttling.

Alarms can be attached to auto scaling, ec2 actions, sns notifications... States: OK, INSUFFICIENT_DATA and ALARM. Period: time to evaluate the metric that triggers the alarm, enable high resolution for faster.

Cloudwatch logs: log aggregate. apps can send logs using the sdk, agents can run agents to send logs to cloudwatch, most of the services send by default. The logs can best archived in S3 or send to elastic search. You can use filters in cloudwatch. Logs are grouped in group and each app is a stream. You can tail using sdk and also define expiration policies. You need to have proper permission in aim to stream logs. logs are encrypted at a group level using kms.

Cloudwatch Events can be schedule: as cron jobs, event pattern rules in cloudwatch. You can call lambda functions, send sns notifications, etc. They use json format.

AWS X-Ray. Distributed tracing, logging and monitoring. Troubleshoot performance, understand dependencies, review requests, bottlenecks, etc. It is compatible with many services. Each trace has several segments, one for each call. You can add annotations to the traces. You can trace every requests or aggregate them. There is IAM access control and KMS encryption at rest. Your code must use X-ray sdk. You need a small code modification not like Istio. The code snippet will capture: AWS calls, http calls, database calls, sqs calls, etc. You need to also install x-ray daemon or enable the AWS integration for x-ray. AWS lambda and other services already run the daemon. You need IAM write rights to send data to x-ray. Make sure that Lambda has the proper role. For EC2 besides the role you need the daemon.

AWS cloud trail provides governance, compliance and audit for your account. It is enabled by default. You get a history in the console from any change in the console, sdk, cli or AWS service. You can put logs from cloud trail into cloud watch. If a resource gets deleted you can find the event in cloud trail and see what happened. Everything you do in AWS is tracked here, great for audits.

AWS has **Amazon ES** which is the elastic search and Kibana offerings that can be used to analize cloud watch logs. Athena big data analysis tool can also query logs.

AWS config is a fully managed service that provides resource inventory, config history, config change notifications and more.

Messaging

SQS for queue model, SNS for pub subscribe and Kinesis for stream processing, these are ways to decouple service calls using async calls, this extra queue layer is great for back pressure, if you get a spike in traffic the queue will hold messages so the receivers will not go down.

SQS: 3 types of queues. Standard: fully managed, oldest offering, scales from 1 to 10.000 messages per second and holds them for 4 days by default and 14 max. Low latency < 10ms, horizontal scaling using consumers. It may be duplicate messages and out of order. Max size message is 256kb. Delay queue: can delay messages up to 15min, default is 0. DelaySeconds parameter. It has a body and then key value properties called meta data. In SQS consumers poll the data and can receive up to 10 message at a time. Consumers must process the messages within the visibility timeout period, the delete the message with the ID and a delete handle function. If there are multiple consumers each one will get the message one by one and the consumer needs to process it within the visibility timeout during this time other consumers will not see it. After the time out they will. the default timeout is 30 sec and the max 12 hours. If the operation is idempotent set it low for performance and if one message takes too long it will get process more than one. If not, you can still use the ChangeMessageVisibility API which allows you to request more time. Dead Letter Queue: 3rd type, failed messages go back to the queue in the standard queue, if the problem is the message a loop will be created, you can set a redrive policy to stop the loop and then use the dead letter queue to send error messages. To avoid many api calls you can enable long poling up to 20 seconds where consumer will wait if there are no messages. Set WaitInSeconds to set the value. They charge per api call so long poling is cheaper. FIFO queue is newer queue not available in all regions, less throughput < 300 but exactly one warranty and order. You can set consumer groups based on content and have automatic de duplication based on the body hash.

For large messages use SQS extended client that leverages on S3, body is encrypted but not metadata, so do not put sensitive data. No VPC end point for SQS, you need internet connectivity. PurgeQueue to delete all messages in queue. There is a batch API you can use to decrease cost.

SNS: pub subscribe. a producer sends a singe message to a topic and many subscribers consume the topic, subscribers can be SQS, http end points, lambda, email, sms, push notifications. It integrates with many services: Cloudwatch, cloud formation, s3, etc. You need mobile sdk to send push notifications. A common pattern is the SNS + SQS fan out where you put a message once in SNS and this is sent to many SQS queues. This is the overcome the problem that SNS doesnt hold the messages and SQS does.

Kinesis is a managed alternative to Kafka, although recently they released KMS managed kafka since it is better than kinesis. Compatible with Spark. It is HA using 3 AZs. It has several APIs like Kafka: streams, analytics (ksql) and firehorse(connect). Retention from 1 to 7 days. You can replay data. You need to define shards(partitions) in advanced and you are billed per shard. You can batch calls. Hot partition problem is when you use a key that is not evenly distributed so most of the messages go to the same partition. You can use SDKs or cli. ProvisionThroughputExceeded is thrown if you sent too many messages, just retry, add more shard or make sure you dont have a hot partition. Besides the SDK you

can use the client library KCL which uses DynamoDB to track the offsets. Data Base64 encoded. You can use Kinesis inside your VPC. Data Analytics: real time sql, can create new streams, you pay for the data you consume, it has auto scaling. Firehose is for ETL, a bit more latency, many data formats but you pay for conversion. You pay for data. SQS and Kinesis consumer pull data and SNS they get push. Kinesis only one consumer per shard. You must provision your throughput in advance for Kinesis not in SQS.

PART 2: SERVERLESS

It is anything that doesn't require managed infrastructure. Started with Lambda and Faas but now it has authentication and storage like DynamoDB. The idea is that you don't deploy services but code as functions, or in DB, you do not manage a database just store data. EC2 you need to provision in advance, they are continuously running, you need to bootstrap them and do other management. You are limited by RAM and CPU.

Lambda are virtual functions that run when called and they automatically scale without intervention, they are short lived and you pay per usage. It integrates with many services and it is the glue between them based on events. Supports many languages and it is integrated with cloud watch for monitoring. You need to define the RAM in advance but it is easy to get and it is based on the code not load. It is very cheap. Functions have roles to define what can they do in the AWS infrastructure.

Configuration: timeout is the max execution time, by default is just 3 seconds. Max 300. You send ENV variables, set RAM, you can deploy in your VPC. Lambda functions can have up to 1000 parallel executions. You can set reserve concurrency to lower the value, once the max is passed throttling takes place. Throttle error is 429. You can run the functions async and they will retry 2 times after that they go to the DLQ queue which can be SNS or SQS. You need to add SNS and SQS permissions to the role to write to the queue. Lambda is integrated with X-ray. You can write to tmp folder up to half GB. If you upload a function it must be a zip file up to 50mb and uncompress 250mb. Functions have versions. \$LATEST is the last one which is mutable, then you publish a immutable version. a version is the code and configuration. Aliases are pointers to lambda versions this way you can have your dev,test, prod environment and point them to different function versions. You can do blue green and canary deployments with lambda and assign certain percent of traffic to different versions. So users interact with the same arn and you can rewire the aliases. Do heavy work like connect to DBs outside the handler otherwise you will re initialize the connections. Use env variables, do not hard code urls. Encrypt passwords with KMS. Break down big functions. do not use recursions and VPC is slower so try to avoid. Lambda invocation type: RequestResponse sync or Event async. Increased RAM will also allocate more CPU and network bandwidth.

The code package is stored in s3 and on first invocation the code is pull and a set of containers are created using cloud formation, this enables hot invocations.

It is common to write lambda functions for ci/CD that do integration test simulating event sources using the invoke api, or functions that update the aliases, this is done thanks to events trigger by code pipeline that can call lambda.

AWS CodeStar is a unified UI for creating serverless apps using best practices. It comes with a pre build pipeline for serverless apps. It comes with member management, issue tracking and operations.

Dynamo DB: NoSQL, no direct supports for joins or aggregations like join, no relational data but distributed horizontal scaling which is great for lambda. By default Dynamo is highly available using 3 AZs. Highly scalable and serverless, you do not manage it. Millions of request per seconds, fast and consistent performance, integrated with IAM. You can do event driven programing using Dynamo Streams. Very low cost and auto scaling.

DynamoDB is made of tables each one with a primary key, each table has infinite number of items (rows), each item has attributes similar to columns but they can be nested and can be added over time, you can have list and maps as attributes. Primary key two options: partition key only, this is a hash like Kinesis, must be evenly distributed. The second option is having a combination of a primary key and sort key, the combination must be unique, data is grouped by partition key and sorted by sort key. You start by creating tables, no need to create or manage a DB.

Tables capacity must be calculated and provision in advance. there is read capacity units RCU and write WCU. You can set up auto scaling and you also have burst credit for peak demand. When the credits expire you get the ProvisionThroughputException, in this case do exponential back off. 1 WCU = 1 write per second for item up to 1KB, if more than 1KB, one extra WCU per KB. In DynamoDB you can select between strong consistency read and eventual. By default is strong to get items but you can change it using ConsistentRead parameter. RCU = 1 strong read per second or 2 eventual per second for 4KB size. WCU and RCU are spread over partitions. If you exceed throughput you get the ProvisionThroughputException which may be caused by a hot partition, large items or hot keys. Solutions: use an evenly distributed key, back off, or use DynamoDB accelerator for read which is a cache. PutItem create or fully replace item. UpdateItem partial update of attributes. Conditional writes allow to write or read info only if a condition is true, this is good for concurrency since you can only replace a value if the original value is what you expected. There is also DeleteItem and DeleteTable. It is much efficient and cheaper to use batch: BatchWriteItem up to 25 items for put or delete item. 16MB max batch, only one API call so less latency and AWS run the batch in parallel. Part of the batch can fail and is up to the client to retry. GetItem: read by key, uses eventual read by default. use ProjectionExpression to retrieve only some of the attributes. BatchGetItem: up to 100 items in parallel. Query returns values based on the key and sort key. You can define a filter as well but this is executed on the client. Up to 1MB of data returned and you can specify a limit. It is very efficient. Scan goes through the whole table and then filters data which is not efficient, consumes a lot of RCU. You can use parallel scan for efficiency but lots of RCU. You can use Projection and Filter Expression. Scan is needed to see the whole table query needs equals in primary key so it will return just one item or group of items in case it is a primary + sort key. You can have up to 5 alternative local secondary indexes which are extra sort keys that must be defined when creating the table. There are also global secondary indexes to speed up queries on non keys attributes, the index is a new table that you can query and project values. GSI can be modified after table creation. So local secondary is to speed up queries of attributes inside a table that you are querying using the defined primary key and global indexes is to query primary keys that are not defined PKs, so you create new tables. DynamoDB is optimistic locking, you have conditional updates and deletes under the hood using version numbers.

DAX is DynamoDB accelerator, super performance read cache. When enabled writes go through DAX. It is multi AZ, up to 10 nodes and default TTL is 5 min. DynamoDB streams are a change log like kafka, every update, create and delete go to the stream. Lambda functions can subscribe to DynamoDB streams to react in real time to events. You can use the streams to replicate data across regions, the retention policy is 24h. DynamoDB supports VPC so you can access without internet and it is fully managed by IAM. You can have global tables across regions using streams as replication. There is a tool called **DMS** to import data from Mongo and other DBs. You can run it locally.

API Gateway is AWS api management tool to expose rest end points. Lambda + API gateway is serverless. It handles versioning of an API to support different versions for clients, authorization and authentication, different environments, swagger ad open api, key management and client throttling. It can also parse transform and validate request, it has a cache and also generates SDK and API specs. You need to make a deployment in the gateway for changes to take effect, you deploy to stages: dev, test, prod. You can have many stages and you have full history for rollback, each stage has env variables to set properties for each environment. This way you can change http end points and lambda functions (using context parameters). A common pattern is to use env variables to point to lambda aliases. API management support canary deployments. Mapping templates are used to modify in and out requests. You can parse the body, parameters, change content, add headers, map json to xml, etc. It uses VLT language.

Swagger is now a standard called Open API which AWS API gateway support. It can be written in yaml or json. You can export or import directly from AWS. Swagger can generate SDK for mobile apps. API gateway can use cache, default 5 min max 1h, clients with the right role can bypass cache using header: Cache-Control:max-age=0.

For monitoring enable cloud watch at stage level. You have the request and response and many metrics, It also integrates with x ray.

If you want to support calls from other rest services you need to enable CORS, in the console you can generate the OPTIONS call with the desired response. In API gateway you can define plans for different clients each one will have an API key, quota per stage and throttling settings. You define user plans first and then associate keys with them to reuse settings. For security create a IAM policy and assign it to a role, the gateway will apply permissions based on the a service called Sig v4 which attached credentials to requests headers. Lambda authorizers previously known as custom authorizers are used from OAuth, SAML and any other 3rd party to validate credentials, it uses a cache. A lambda function is executed which returns the policy for the given credentials. There is also cognito users pools for oauth. So, IAM is for users and apps within AWS, it handles authentication and authorization. Custom authorizer is for 3rd party tokens, very flexible and does auth plus authorization but you pay for lambda function call. Cognito is great for authentication using oauth google, facebook, etc. it does only authentication and you have to implement authorization.

Cognito is the authentication module for AWS. It has the Cognito User Pools (**CUP**) which is a serverless DB of user identities that can be simple(username and password) or federated using OAuth like Facebook or Google; or SAML. It is based on JWT and integrated with API gateway which uses the JWT. There are also Federated Identity pools which are used to get temp access to AWS resources using identity providers. So you can login with facebook, get a token the request the federated pool for temp access to a S3 bucket if granted the client gets this policy for a time period. This is for direct access while the user pool is for permanent using api gateway. Cognito Sync is deprecated for AppSync, it is used to store config, state and preferences of applications, it has cross device sync (android or iOS) and offline mode. It relies on federated pool.

SAM is the serverless application model. It is a complete framework for development and deployment of serverless apps. It uses SAM Yaml file that is used to generate complex cloud formation deployments. Only two commands to deploy to Prod and you can use CodeDeploy. With SAM you can run lambda, API gateway and DynamoDB locally. The SAM template is a yaml file that starts with Transform: AWS::Serverless. It has 3 core sections AWS::Serverless:Function for lambda,

AWS::Serverless::api for API gateway and AWS::Serverless::simple table for dynamo. To deploy you can use SAM package and deploy or aws cloudformation package and deploy for cli.

KMS is AWS key management service which manages all the keys for data encryption. It is fully integrated with IAM. KMS is integrated with many services for encryption. You can also use the SDK and cli to encrypt and decrypt data. KMS manages your private key, it can rotate it for extra security, you do not need to worry about managing the key, you cannot even retrieve your own private key used for encryption, you can only ask KMS to encrypt data. The user master key is called CMK. So, you never store your secrets in plaintext, you ask KMS to encrypt them and then store them in env variables. It can only encrypt 4k data per call. For bigger use envelope encryption.

To give access to KMS to a user make sure the key policy allows the user and that the IAM role allow the KMS API calls. KMS give us full key and policy management and cloud trail for audit.

Three types of CMK, the default master which is free, you can create your own in KMS for 1\$ a month or create your own and import it for 1\$ month. There is also a charge per call to KMS. KMS is under IAM in the console under encryption keys. The key policy describes who can use the key and what they can do like update or delete. Encryption SDK is provided to do envelope encryption for data bigger than 4KB. This is different from S3 encryption SDK. The API call to encrypt is GenerateDataKeyAPI.

AWS parameter store **SSM** is used to store secrets and configuration. It uses KMS and it is serverless so you don't have to do anything and you have a secure vault. There is a SDK and free, there is also version tracking to recover. The configuration management uses IAM, it integrates with cloud watch events for notifications and cloudformation for deployments with secrets. So SSM is an easier alternative when you want to store parameters than using KMS directly. It uses a tree structure so you can define by app, then environment, etc. In the console is under EC2 or systems management. The parameters are based on paths so you can search by path: myapp/dev/dbpasswd.

Cross account access: define a role in IAM for another account, define which accounts can access, use AWS **STS** (secure token service) to get a token so the external account can assume the role using AssumeRole API, the access is between 15 min and 1h.

Cloud Front is an AWS CDN to add cache in the current 168 edge locations across regions, designed for S3 because a bucket can be created only in one region but also works for EC2. Can protect against DDoS attacks and it has SSL termination but also can talk to your services using ssl. It supports RTMP for video.

Step functions are used to create a state machine using a JSON file to orchestrate several lambda functions calls. There is also a designer. It integrates with EC2, ECS, api gateway.... You can have also human approval. It is used for workflows.

SWF is similar to step functions but for EC2, not serverless, it is the simple workflow service. The manual intervention is better and supports parent and child processes, other than that AWS recommends step functions.

ECS is the container orchestration solution for AWS. There is also has EKS which is kubernetes whereas ECS is amazon specific Docker orchestration. The Core service runs containers on your user defined EC2 instances, simple: just runs Docker on linux EC2. **Fargate** is another solution to run tasks(pods) on containers on AWS managed instances, so it is serverless. **EKS** is Kubernetes managed by AWS, more portable. **ECR** is the docker registry managed by Amazon. ECS + ECR is great alternative to microservices if you don't use lambda. Lambda is more popular for green field and ECS to bring on prem services. Also ECS is popular for CI and batch. You create an ECS cluster on EC2 instances and define services which are replicated and call tasks(pods) which are running containers. Each EC2 can have multiple services and tasks, each container has integration with IAM for security. The new ALB load balancer has integration with ECS for port mapping so you can replicate containers and assign dynamic ports so they can run on the same instance. This allows max utilization of cpu and ram and easier rolling updates. You need to run ECS agent with its config file on each EC2 instance. On EC2 Linux the agent is already installed but you still need to modify the ecs.config file. Main properties: ECS_CLUSTER, ECS_ENGINE_AUTH_DATA to authenticate to external repositories, ECS_AVAILABLE_LOGGING_DRIVERS to use cloud watch for logs, ECS_ENABLE_TASK_IAM_ROLE to enable role and policy check when running containers, very important, set it to true. ECR is the image repo fully integrated with IAM and secure. Code Build can build and push images to ECR.

SES is the simple email server. Integrated with IAM for security and with many services like S3 and Lambda. You can use SMTP or the SDK.

Redshift is the big data service for big queries, analytics and OLAP which is slow big data queries. This is the data warehouse and data lake solution. Neptune is the graph DB in AWS. DMS is the AWS sqoop equivalent, used to migrate data between RDS, DynamoDB and specially Redshift. There are other migration tools.

12. IAM Introduction

IAM 101 Brain Dump

- One IAM User per PHYSICAL PERSON
- One IAM Role per Application
- IAM credentials should NEVER BE SHARED
- Never; ever; ever; ever; write IAM credentials in code. EVER.
- And even less, NEVER EVER EVER COMMIT YOUR IAM credentials
- Never use the ROOT account except for initial setup.
- Never use ROOT IAM Credentials

EC2

15. EC2 Introduction

What is EC2?

- EC2 is one of most popular of AWS offering
- It mainly consists in the capability of :
 - Renting virtual machines (EC2)
 - Storing data on virtual drives (EBS)
 - Distributing load across machines (ELB)
 - Scaling the services using an auto-scaling group (ASG)
- Knowing EC2 is fundamental to understand how the Cloud works



Amazon EC2



AMI -> Amazon Machine Image

Price Comparison Example – m4.large – us-east-1

Price Type	Price (per hour)
On-demand	\$0.10
Spot Instance (Spot Price)	\$0.032 - \$0.450 (up to 90% off)
Spot Block (1 to 6 hours)	~ Spot Price
Reserved Instance (12 months) – no upfront	\$0.062
Reserved Instance (12 months) – all upfront	\$0.058
Reserved Instance (36 months) – no upfront	\$0.043
Reserved Convertible Instance (12 months) – no upfront	\$0.071
Reserved Dedicated Instance (12 months) – all upfront	\$0.064
Reserved Scheduled Instance (recurring schedule on 12 months term)	\$0.090 – \$0.095 (5%-10% off)
Dedicated Host	On-demand price
Dedicated Host Reservation	Up to 70% off



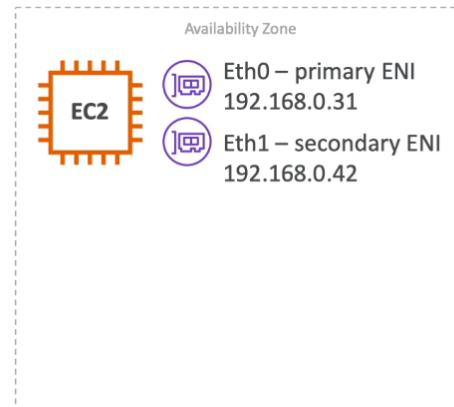
© Stephane Maarek

CC BY-SA

www.datacumulus.com

Elastic Network Interfaces (ENI)

- Logical component in a VPC that represents a virtual network card
- The ENI can have the following attributes:
 - Primary private IPv4, one or more secondary IPv4
 - One Elastic IP (IPv4) per private IPv4
 - One Public IPv4
 - One or more security groups
 - A MAC address
- You can create ENI independently and attach them on the fly (move them) on EC2 instances for failover
- Bound to a specific availability zone (AZ)



© Stephane Maarek

Das heißt, wenn Sie eine ENI in der bestimmten AZ erstellen,

-by-Stephane Maarek

33. High Availability and Scalability

Vertical Scalability

- Vertically scalability means increasing the size of the instance
- For example, your application runs on a t2.micro
- Scaling that application vertically means running it on a t2.large
- Vertical scalability is very common for non distributed systems, such as a database.
- RDS, ElastiCache are services that can scale vertically.
- There's usually a limit to how much you can vertically scale (hardware limit)



197 Personen haben hier eine Notiz hinzugefügt.

Grenzen für die vertikale

© Stephane Maarek

CC BY-SA

1x 1:39 / 5:05

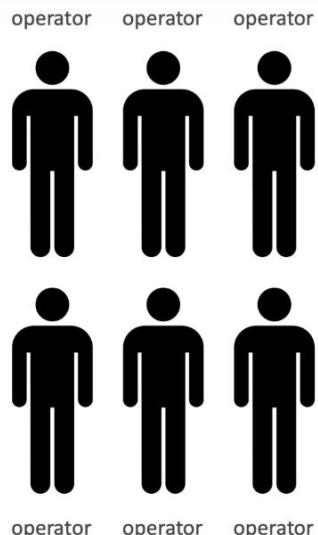
Horizontal Scalability

- Horizontal Scalability means increasing the number of instances / systems for your application
- Horizontal scaling implies distributed systems.
- This is very common for web applications / modern applications
- It's easy to horizontally scale thanks the cloud offerings such as Amazon EC2

197 Personen haben hier eine Notiz hinzugefügt.

© Stephane Maarek

heutzutage einfach, horizontal zu skalieren, da wir



du sahst

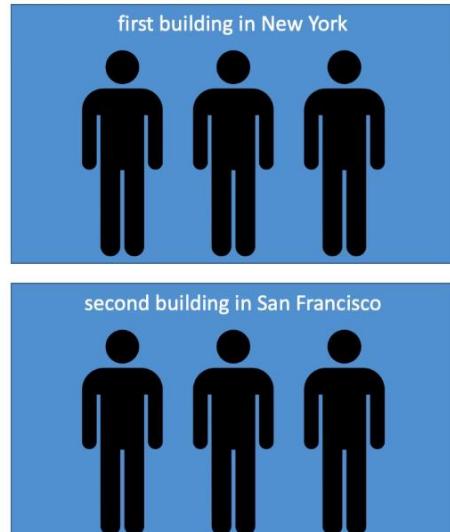
High Availability

- High Availability usually goes hand in hand with horizontal scaling
- High availability means running your application / system in at least 2 data centers (== Availability Zones)
- The goal of high availability is to survive a data center loss
- The high availability can be passive (for RDS Multi AZ for example)
- The high availability can be active (for horizontal scaling)

143 Personen haben hier eine Notiz hinzugefügt.

© Stephane Maarek

alle meine Anrufe in zwei Gebäuden in New York habe.



du sahst

High Availability & Scalability For EC2

- Vertical Scaling: Increase instance size (= scale up / down)
 - From: t2.nano - 0.5G of RAM, 1 vCPU
 - To: u-12tbl.metal – 12.3 TB of RAM, 448 vCPUs
- Horizontal Scaling: Increase number of instances (= scale out / in)
 - Auto Scaling Group
 - Load Balancer
- High Availability: Run instances for the same application across multi AZ
 - Auto Scaling Group multi AZ
 - Load Balancer multi AZ

Sie müssen verstanden werden, wenn Sie

© Stephane Maarek

34. Elastic Load Balancing (ELB) Overview

What is load balancing?



- Load balancers are servers that forward internet traffic to multiple servers (EC2 Instances) downstream.



Why use a load balancer?

- Spread load across multiple downstream instances
- Expose a single point of access (DNS) to your application
- Seamlessly handle failures of downstream instances
- Do regular health checks to your instances
- Provide SSL termination (HTTPS) for your websites
- Enforce stickiness with cookies
- High availability across zones
- Separate public traffic from private traffic

© Stephane Maarek

Load Balancer in Ihre EC2-Instanzen gelangen.

JU.Schule

Why use an EC2 Load Balancer?

- An ELB (EC2 Load Balancer) is a **managed load balancer**
 - AWS guarantees that it will be working
 - AWS takes care of upgrades, maintenance, high availability
 - AWS provides only a few configuration knobs
- It costs less to setup your own load balancer but it will be a lot more effort on your end.
- It is integrated with many AWS offerings / services

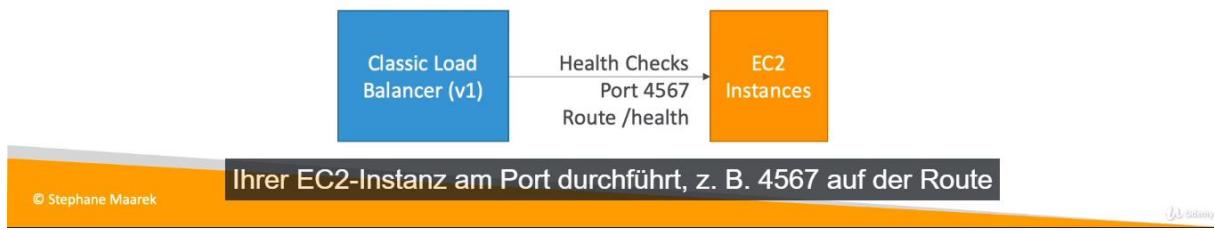
© Stephane Maarek

Verwenden Sie in AWS ständig Load Balancer.

JU.Schule

Health Checks

- Health Checks are crucial for Load Balancers
- They enable the load balancer to know if instances it forwards traffic to are available to reply to requests
- The health check is done on a port and a route (`/health` is common)
- If the response is not 200 (OK), then the instance is unhealthy



Types of load balancer on AWS



- AWS has [3 kinds of managed Load Balancers](#)
- Classic Load Balancer (v1 - old generation) – 2009
 - HTTP, HTTPS, TCP
- Application Load Balancer (v2 - new generation) – 2016
 - HTTP, HTTPS, WebSocket
- Network Load Balancer (v2 - new generation) – 2017
 - TCP, TLS (secure TCP) & UDP
- Overall, it is recommended to use the newer / v2 generation load balancers as they provide more features
- You can setup [internal](#) (private) or [external](#) (public) ELBs



34. Elastic Load Balancing (ELB) Overview

Load Balancer Security Groups



Load Balancer Security Group:

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	0.0.0.0/0	Allow HTTP from an...
HTTPS	TCP	443	0.0.0.0/0	Allow HTTPS from a...

Application Security Group: Allow traffic only from Load Balancer

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	sg-054b5ff5ea02f2b6e (load-b	Allow Traffic only...

Ich hoffe, Sie sehen,

34. Elastic Load Balancing (ELB) Overview

Load Balancer Good to Know

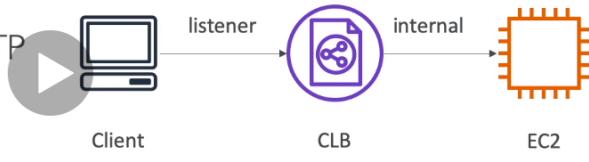
Drucken Sie Esc, um den Vollbildmodus zu verlassen

- LBs can scale but not instantaneously – contact AWS for a “warm-up”
- Troubleshooting
 - 4xx errors are client induced errors
 - 5xx errors are application induced errors
 - Load Balancer Errors 503 means at capacity or no registered target
 - If the LB can't connect to your application, check your security groups!
- Monitoring
 - ELB access logs will log all access requests (so you can debug per request)
 - CloudWatch Metrics will give you aggregate statistics (ex: connections count)

Das ist es.

Classic Load Balancers (v1)

- Supports TCP (Layer 4), HTTP & HTTPS (Layer 7)
- Health checks are TCP or HTTP based
- Fixed hostname
XXX.region.elb.amazonaws.com



© Stephane Maarek

Schauen wir uns also an, wie wir dies in der Konsole tun können.

Überblick

Application Load Balancer (v2)



- Application load balancers is Layer 7 (HTTP)
- Load balancing to multiple HTTP applications across machines (target groups)
- Load balancing to multiple applications on the same machine (ex: containers)
- Support for HTTP/2 and WebSocket
- Support redirects (from HTTP to HTTPS for example)

© Stephane Maarek

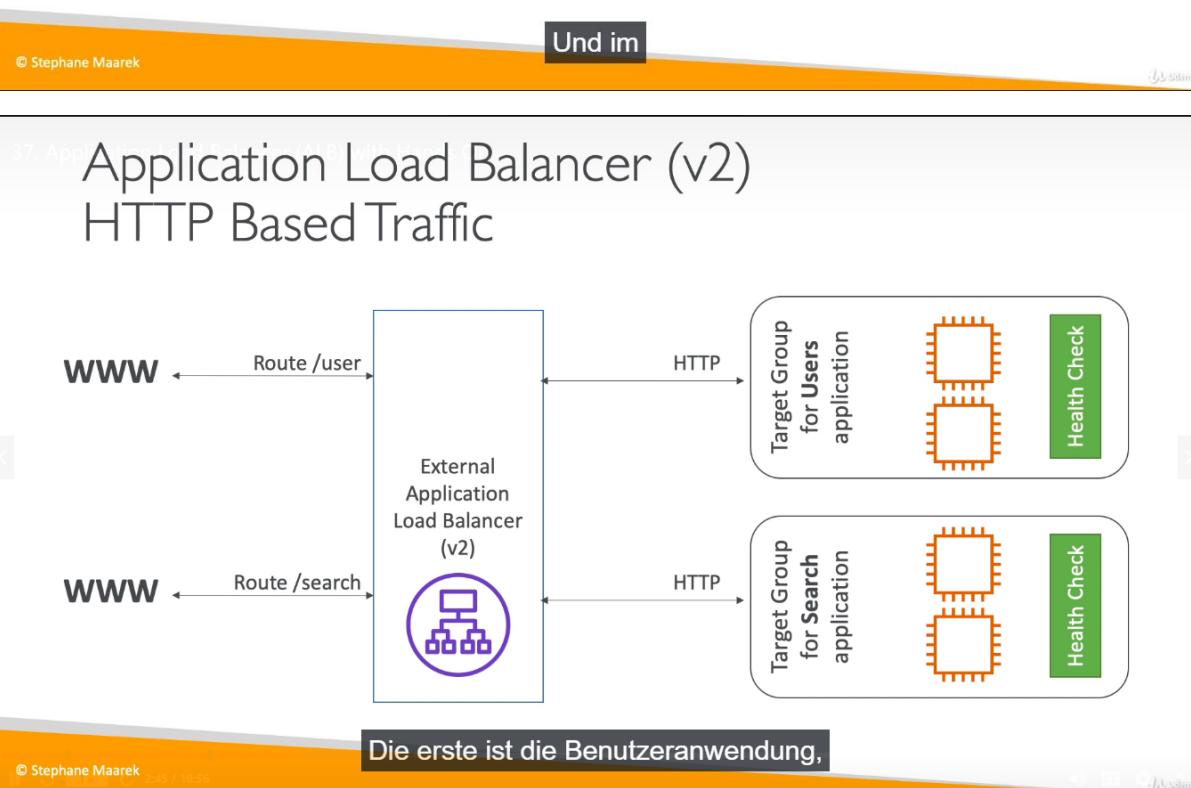
zu HTTPS umleiten möchten, kann dies auf Load-Balancer-Ebene erfolgen.

Überblick

Application Load Balancer (v2)



- Routing tables to different target groups:
 - Routing based on path in URL (example.com/users & example.com/posts)
 - Routing based on hostname in URL (one.example.com & other.example.com)
 - Routing based on Query String, Headers (example.com/users?id=123&order=false)
- ALB are a great fit for micro services & container-based application (example: Docker & Amazon ECS)
- Has a port mapping feature to redirect to a dynamic port in ECS
- In comparison, we'd need multiple Classic Load Balancer per application



Application Load Balancer (v2)

Target Groups

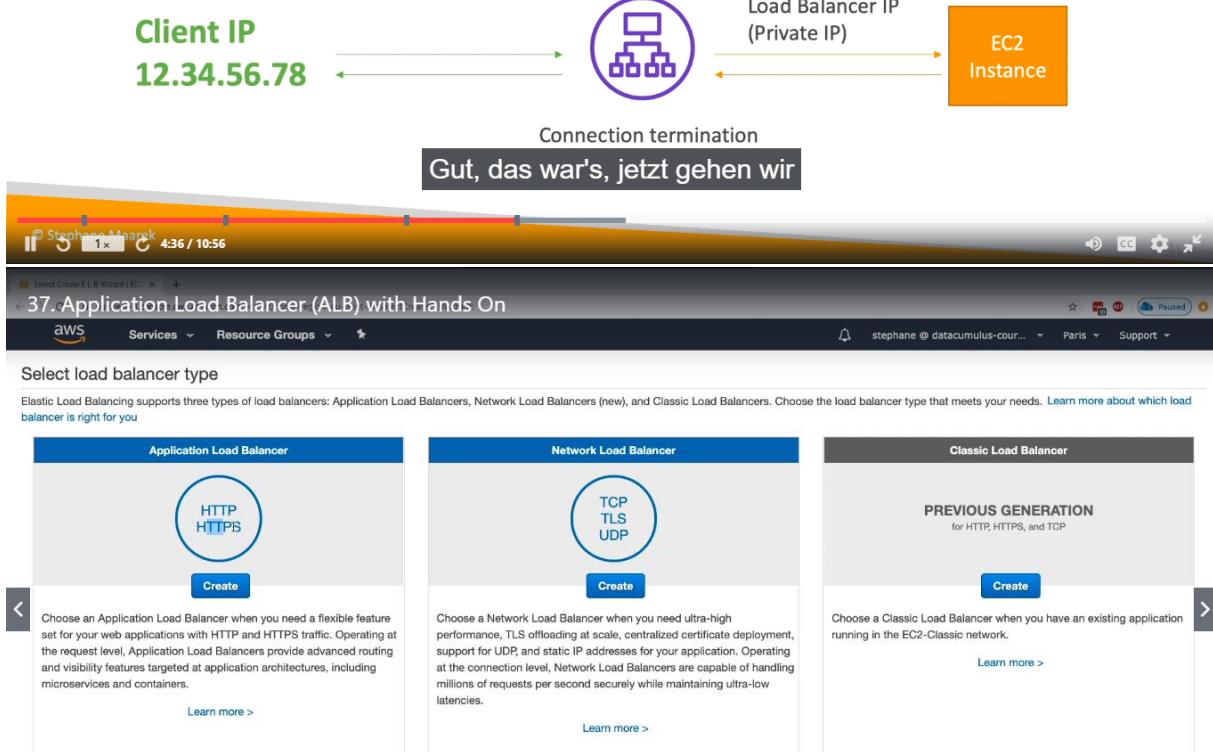
- EC2 instances (can be managed by an Auto Scaling Group) – HTTP
 - ECS tasks (managed by ECS itself) – HTTP
 - Lambda functions – HTTP request is translated into a JSON event
 - IP Addresses – must be private IPs
-
- ALB can route to multiple target groups
 - Health checks are at the target group level

37. Application Load Balancer (ALB) with Hands On

Application Load Balancer (v2)

Good to Know

- Fixed hostname (`XXX.region.elb.amazonaws.com`)
 - The application servers don't see the IP of the client directly
 - The true IP of the client is inserted in the header `X-Forwarded-For`
 - We can also get Port (`X-Forwarded-Port`) and proto (`X-Forwarded-Proto`)



um Application Load Balancer, der nur für HTTPS- und HTTP-Verkehr vorgesehen ist.

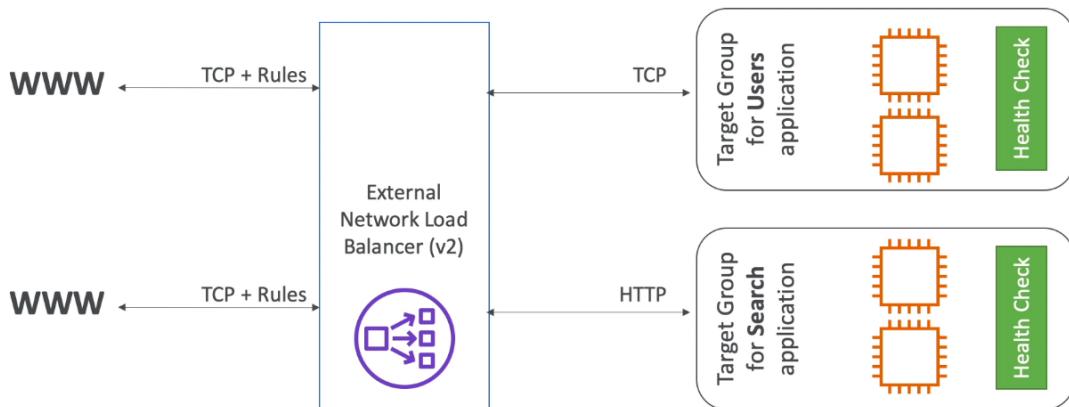
Network Load Balancer (v2)



- Network load balancers (Layer 4) allow to:
 - Forward TCP & UDP traffic to your instances
 - Handle millions of requests per second
 - Less latency ~100 ms (vs 400 ms for ALB)
- NLB has one static IP per AZ, and supports assigning Elastic IP (helpful for whitelisting specific IP)
- NLB are used for extreme performance, TCP or UDP traffic
- Not included in the AWS free tier

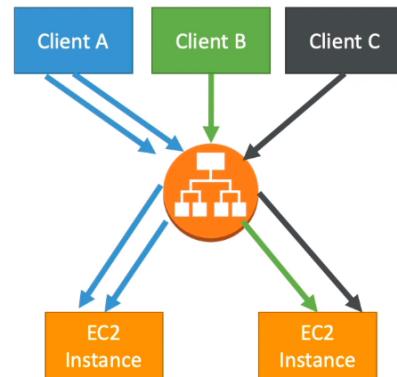


Network Load Balancer (v2) TCP (Layer 4) Based Traffic



Load Balancer Stickiness

- It is possible to implement stickiness so that the same client is always redirected to the same instance behind a load balancer
- This works for Classic Load Balancers & Application Load Balancers
- The “cookie” used for stickiness has an expiration date you control
- Use case: make sure the user doesn’t lose his session data
- Enabling stickiness may bring imbalance to the load over the backend EC2 instances



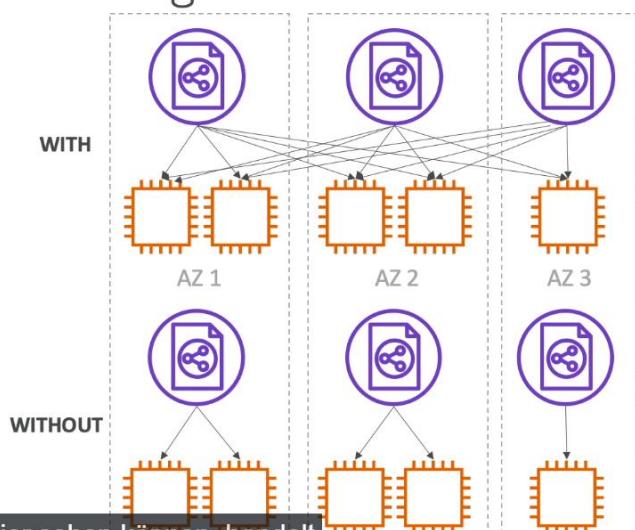
© Stephane Maarek

Lassen Sie uns sehen, wie das in der

Cloud

Cross-Zone Load Balancing

- With Cross Zone Load Balancing; each load balancer instance distributes evenly across all registered instances in all AZ
- Otherwise, each load balancer node distributes requests evenly across the registered instances in its Availability Zone only.



© Stephane Maarek

Da wir hier sehen können, handelt

es sich um

Cross-Zone Load Balancing

- Classic Load Balancer
 - Disabled by default
 - No charges for inter AZ data if enabled
- Application Load Balancer
 - Always on (can't be disabled)
 - No charges for inter AZ data
- Network Load Balancer
 - Disabled by default
 - You pay charges (\$) for inter AZ data if enabled

© Stephane Maarek

die Daten über AZ übertragen werden.

by Sunny

SSL/TLS - Basics

- An SSL Certificate allows traffic between your clients and your load balancer to be encrypted in transit (in-flight encryption)
- SSL refers to Secure Sockets Layer, used to encrypt connections
- TLS refers to Transport Layer Security, which is a newer version
- Nowadays, TLS certificates are mainly used, but people still refer as SSL
- Public SSL certificates are issued by Certificate Authorities (CA)
- Comodo, Symantec, GoDaddy, GlobalSign, DigiCert, LetsEncrypt, etc...
- SSL certificates have an expiration date (you set) and must be renewed

© Stephane Maarek

dass sie authentisch sind, okay?

by Sunny

Load Balancer - SSL Certificates



- The load balancer uses an X.509 certificate (SSL/TLS server certificate)
- You can manage certificates using ACM (AWS Certificate Manager)
- You can create/upload your own certificates alternatively
- HTTPS listener:
 - You must specify a default certificate
 - You can add an optional list of certs to support multiple domains
 - Clients can use SNI (Server Name Indication) to specify the hostname they reach
 - Ability to specify a security policy to support older versions of SSL / TLS (legacy clients)



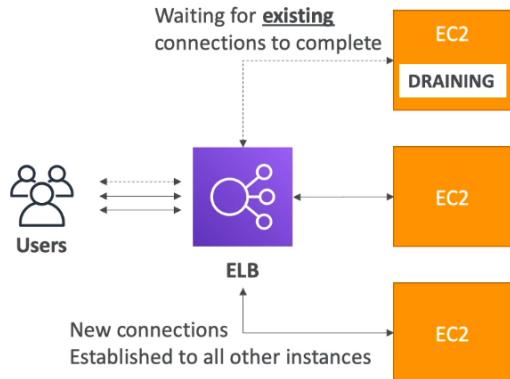
Elastic Load Balancers – SSL Certificates

- Classic Load Balancer (v1)
 - Supports only one SSL certificate
 - Must use multiple CLB for multiple hostname with multiple SSL certificates
- Application Load Balancer (v2)
 - Supports multiple listeners with multiple SSL certificates
 - Uses Server Name Indication (SNI) to make it work
- Network Load Balancer (v2)
 - Supports multiple listeners with multiple SSL certificates
 - Uses Server Name Indication (SNI) to make it work



ELB – Connection Draining

- Feature naming:
 - CLB: Connection Draining
 - Target Group: Deregistration Delay (for ALB & NLB)
- Time to complete “in-flight requests” while the instance is de-registering or unhealthy
- Stops sending new requests to the instance which is de-registering
- Between 1 to 3600 seconds, default is 300 seconds
- Can be disabled (set value to 0)
- Set to a low value if your requests are short



© Stephane Maarek

Wenn Sie also eine Webanwendung

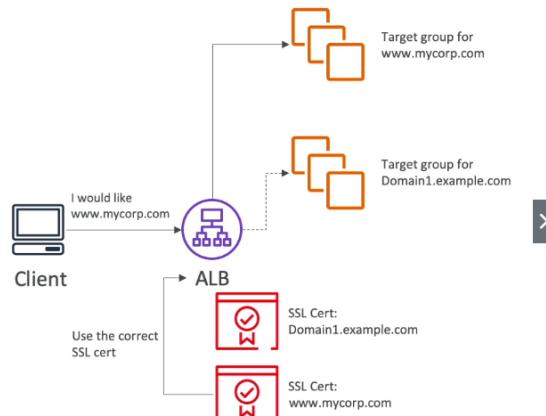
41. Elastic Load Balancer - SSL Certificates

SSL – Server Name Indication

- SNI solves the problem of loading multiple SSL certificates onto one web server (to serve multiple websites)
- It's a “newer” protocol, and requires the client to indicate the hostname of the target server in the initial SSL handshake
- The server will then find the correct certificate, or return the default one

Note:

- Only works for ALB & NLB (newer generation), CloudFront
- Does not work for CLB (older gen)



können Sie das richtige SSL-Zertifikat

...
II Stephane Maarek 1x 5:14 / 7:49

What's an Auto Scaling Group?



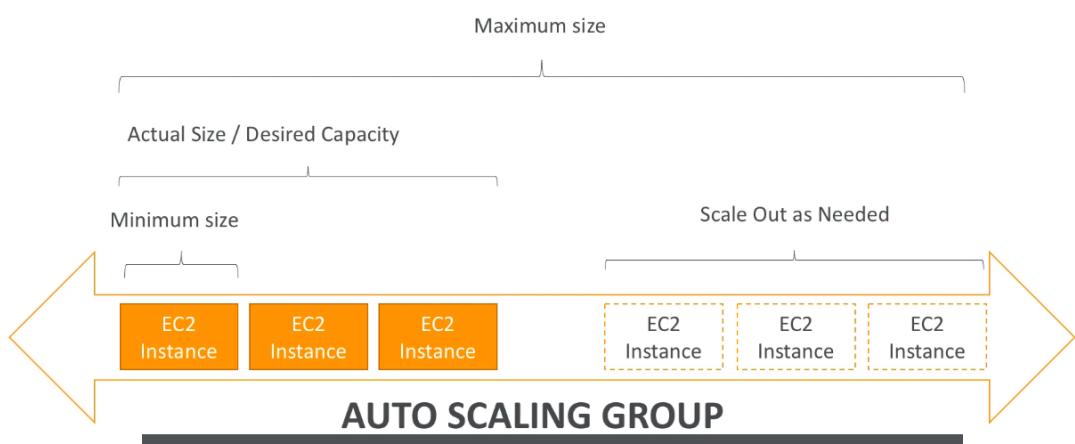
- In real-life, the load on your websites and application can change
- In the cloud, you can create and get rid of servers very quickly
- The goal of an Auto Scaling Group (ASG) is to:
 - Scale out (add EC2 instances) to match an increased load
 - Scale in (remove EC2 instances) to match a decreased load
 - Ensure we have a minimum and a maximum number of machines running
 - Automatically Register new instances to a load balancer

aber natürlich gibt es immer eine Art von Automatisierung,

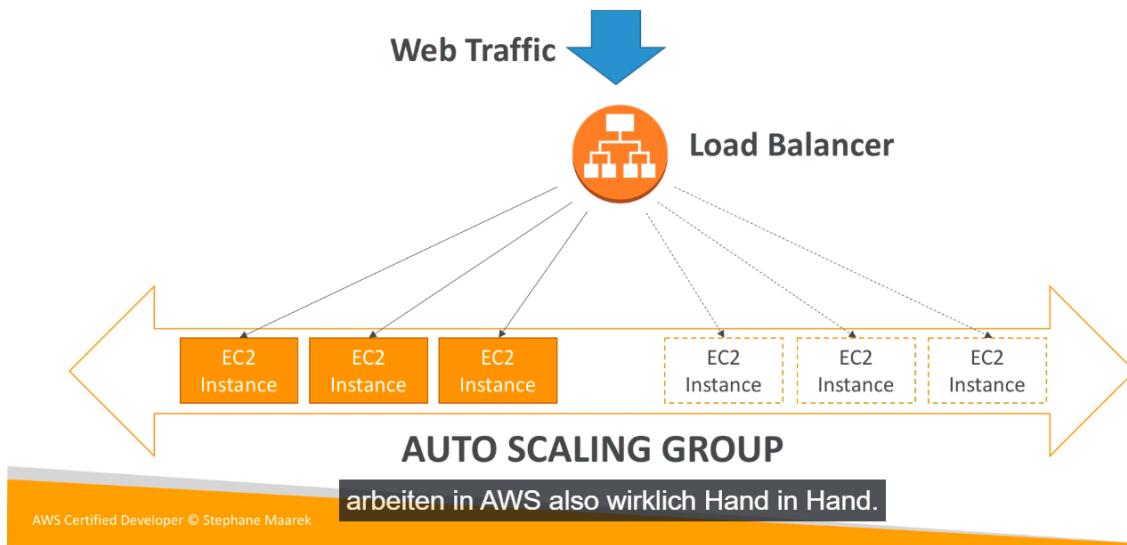
AWS Certified Developer © Stephan Mäurer

by Sunny

Auto Scaling Group in AWS



Auto Scaling Group in AWS With Load Balancer



ASGs have the following attributes

- A launch configuration
 - AMI + Instance Type
 - EC2 User Data
 - EBS Volumes
 - Security Groups
 - SSH Key Pair
- Min Size / Max Size / Initial Capacity
- Network + Subnets Information
- Load Balancer Information
- Scaling Policies

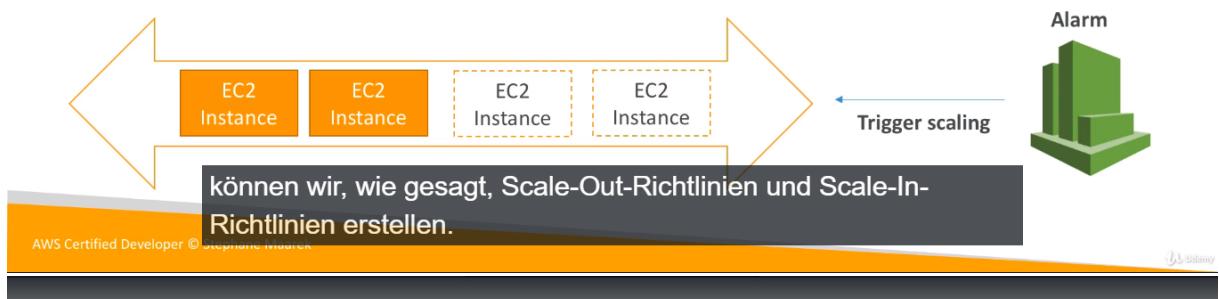
darauf, welchen Load Balancer wir verwenden.

AWS Certified Developer © Stephane Maarek

by

Auto Scaling Alarms

- It is possible to scale an ASG based on CloudWatch alarms
- An Alarm monitors a metric (such as Average CPU)
- Metrics are computed for the overall ASG instances
- Based on the alarm:
 - We can create scale-out policies (increase the number of instances)
 - We can create scale-in policies (decrease the number of instances)



43. Auto Scaling Groups (ASG) Overview

Auto Scaling New Rules

- It is now possible to define "better" auto scaling rules that are directly managed by EC2
 - Target Average CPU Usage
 - Number of requests on the ELB per instance
 - Average Network In
 - Average Network Out
- These rules are easier to set up and can make more sense

Oder "Ich möchte, dass meine CPU-Auslastung durchschnittlich 40% beträgt. Jetzt können Sie die automatische Skalierung



Auto Scaling Custom Metric

- We can auto scale based on a custom metric (ex: number of connected users)
- 1. Send custom metric from application on EC2 to CloudWatch (PutMetric API)
- 2. Create CloudWatch alarm to react to low / high values
- 3. Use the CloudWatch alarm as the scaling policy for ASG



ASG Brain Dump

- Scaling policies can be on CPU, Network... and can even be on custom metrics or based on a schedule (if you know your visitors patterns)
- ASGs use Launch configurations or Launch Templates (newer)
- To update an ASG, you must provide a new launch configuration / launch template
- IAM roles attached to an ASG will get assigned to EC2 instances
- ASG are free. You pay for the underlying resources being launched
- Having instances under an ASG means that if they get terminated for whatever reason, the ASG will automatically **create new ones as a replacement**. Extra safety!
- ASG can terminate instances marked as unhealthy by an LB (and hence replace them)



Auto Scaling Groups – Scaling Policies

- Target Tracking Scaling
 - Most simple and easy to set-up
 - Example: I want the average ASG CPU to stay at around 40%
- Simple / Step Scaling
 - When a CloudWatch alarm is triggered (example CPU > 70%), then add 2 units
 - When a CloudWatch alarm is triggered (example CPU < 30%), then remove 1
- Scheduled Actions
 - Anticipate a scaling based on known usage patterns
 - Example: increase the min capacity to 10 at 5 pm on Fridays

Das ist also die Art von Zeitplanaktionen, die Sie haben würden, nichts drin.

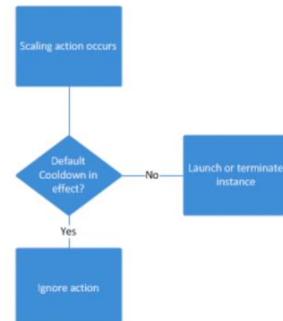
© Stephane Maarek

45 / 100

Scaling Policies

Auto Scaling Groups - Scaling Cooldowns

- The cooldown period helps to ensure that your Auto Scaling group doesn't launch or terminate additional instances before the previous scaling activity takes effect.
- In addition to default cooldown for Auto Scaling group, we can create cooldowns that apply to a specific simple scaling policy
- A scaling-specific cooldown period overrides the default cooldown period.
- One common use for scaling-specific cooldowns is with a scale-in policy—a policy that terminates instances based on a specific criteria or metric. Because this policy terminates instances, Amazon EC2 Auto Scaling needs less time to determine whether to terminate additional instances.
- If the default cooldown period of 300 seconds is too long—you can reduce costs by applying a scaling-specific cooldown period of 180 seconds to the scale-in policy.
- If your application is scaling up and down multiple times each hour, modify the Auto Scaling Groups cool-down timers and the CloudWatch Alarm Period that triggers the scale in



Schauen wir uns nun diese Tötungsrichtlinien direkt in der Konsole an.

© Stephane Maarek

45 / 100

Scaling Policies

What's an EBS Volume?

- An EC2 machine loses its root volume (main drive) when it is manually terminated.
- Unexpected terminations might happen from time to time (AWS would email you)
- Sometimes, you need a way to store your instance data somewhere
- An [EBS \(Elastic Block Store\) Volume](#) is a [network](#) drive you can attach to your instances while they run
- It allows your instances to persist data



© Stephane Maarek

Datenbankdaten auf Ihrem EBS-Volume zu platzieren.

by [Stephane Maarek](#)

EBS Volume

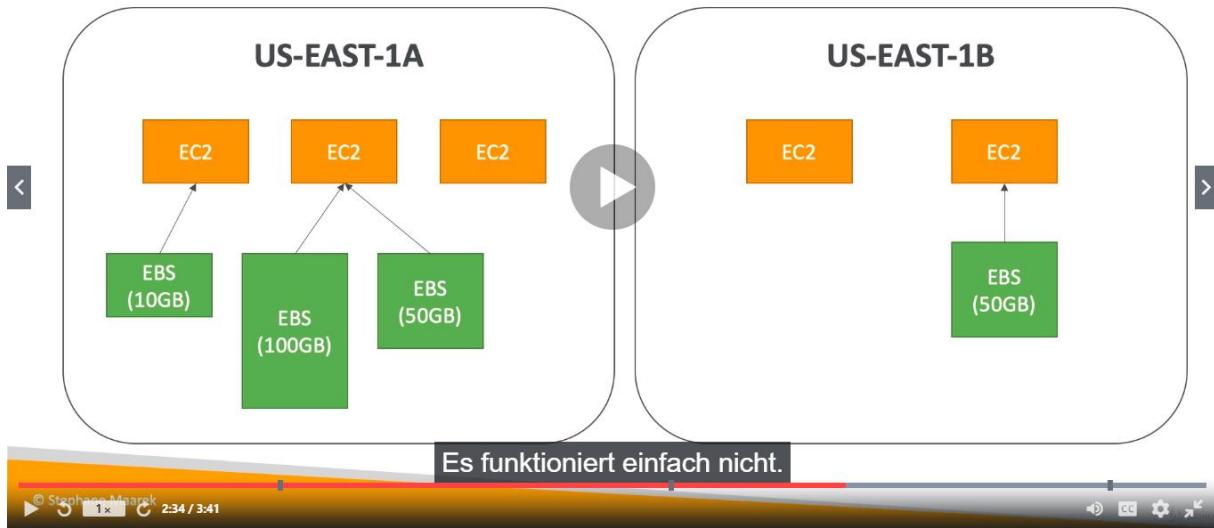
- It's a network drive (i.e. not a physical drive)
 - It uses the network to communicate the instance, which means there might be a bit of latency
 - It can be detached from an EC2 instance and attached to another one quickly
- It's locked to an Availability Zone (AZ)
 - An EBS Volume in us-east-1a cannot be attached to us-east-1b
 - To move a volume across, you first need to snapshot it
- Have a provisioned capacity (size in GBs, and IOPS)
 - You get billed for all the provisioned capacity
 - You can increase the capacity of the drive over time

© Stephane Maarek

Und das ist super wichtig, dass du verstehst.

by [Stephane Maarek](#)

46. EBS Intro EBS Volume Example



EBS Volume Types

- EBS Volumes come in 4 types
 - **GP2 (SSD)**: General purpose SSD volume that balances price and performance for a wide variety of workloads
 - **IO1 (SSD)**: Highest-performance SSD volume for mission-critical low-latency or high-throughput workloads
 - **ST1 (HDD)**: Low cost HDD volume designed for frequently accessed, throughput-intensive workloads
 - **SCI (HDD)**: Lowest cost HDD volume designed for less frequently accessed workloads
- EBS Volumes are characterized in Size | Throughput | IOPS (I/O Ops Per Sec)
- When in doubt always consult the AWS documentation – it's good!
- Only GP2 and IO1 can be used as boot volumes

EBS Volume Types Use cases

GP2 (from AWS doc)

- Recommended for most workloads
 - System boot volumes
 - Virtual desktops
 - Low-latency interactive apps
 - Development and test environments
-
- 1 GiB - 16 TiB
 - Small gp2 volumes can burst IOPS to 3000
 - Max IOPS is 16,000...
 - 3 IOPS per GB, means at 5,334GB we are at the max IOPS

Gig- (stammt) ein GP2-Volume von 5334 Gigabyte haben, sind Sie

© Stephane Maarek

bbb.commy

EBS Volume Types Use cases

IO1 (from AWS doc)

- Critical business applications that require sustained IOPS performance, or more than 16,000 IOPS per volume (gp2 limit)
- Large database workloads, such as:
- MongoDB, Cassandra, Microsoft SQL Server, MySQL, PostgreSQL, Oracle

- 4 GiB - 16 TiB
- IOPS is provisioned (PIOPS) – MIN 100 - MAX 64,000 (Nitro instances) else MAX 32,000 (other instances)
- The maximum ratio of provisioned IOPS to requested volume size (in GiB) is 50:1

Schauen wir uns jetzt an, wie das in der Konsole ist.

© Stephane Maarek

bbb.commy

EBS Volume Types Use cases

ST1 (from AWS doc)

- Streaming workloads requiring consistent, fast throughput at a low price.
 - Big data, Data warehouses, Log processing
 - Apache Kafka
 - Cannot be a boot volume
-
- 500 GiB - 16 TiB
 - Max IOPS is 500
 - Max throughput of 500 MiB/s – can burst

noch viele Durchsätze von bis zu 500 Megabyte pro Sekunde erzielen

© Stephane Maarek

bbz salzburg

EBS Volume Types Use cases

SCI (from AWS doc)

- Throughput-oriented storage for large volumes of data that is infrequently accessed
 - Scenarios where the lowest storage cost is important
 - Cannot be a boot volume
-
- 500 GiB - 16 TiB
 - Max IOPS is 250
 - Max throughput of 250 MiB/s – can burst

Burst-Fähigkeit, so dass es wie ein weniger guter st1 ist.

© Stephane Maarek

bbz salzburg

EBS – Volume Types Summary

- gp2: General Purpose Volumes (cheap)
 - 3 IOPS / GiB, minimum 100 IOPS, burst to 3000 IOPS, max 16000 IOPS
 - 1 GiB – 16 TiB , +1 TB = +3000 IOPS
- io1: Provisioned IOPS (expensive)
 - Min 100 IOPS, Max 64000 IOPS (Nitro) or 32000 (other)
 - 4 GiB - 16 TiB. Size of volume and IOPS are independent
- st1: Throughput Optimized HDD
 - 500 GiB – 16 TiB , 500 MiB /s throughput
- sc1: Cold HDD, Infrequently accessed data
 - 250 GiB – 16 TiB , 250 MiB /s throughput

© Stephane Maarek

Wir sehen uns in der nächsten Vorlesung.

bj.Sunny

EBS vs Instance Store

- Some instance do not come with Root EBS volumes
- Instead, they come with “Instance Store” (= ephemeral storage)
- Instance store is physically attached to the machine (EBS is a network drive)
- Pros:
 - Better I/O performance
 - Good for buffer / cache / scratch data / temporary content
 - Data survives reboots
- Cons:
 - On stop or termination, the instance store is lost
 - You can't resize the instance store
 - Backups must be operated by the user

© Stephane Maarek

also einen großen Anwendungsfall für Caches oder was auch immer, aber sie

bj.Sunny

Local EC2 Instance Store

- Physical disk attached to the physical server where your EC2 is
- Very High IOPS (because physical)
- Disks up to 7.5 TiB (can change over time), striped to reach 30 TiB (can change over time...)
- Block Storage (just like EBS)
- Cannot be increased in size
- Risk of data loss if hardware fails

Instance Size	100% Random Read IOPS	Write IOPS	Very high IOPS
i3.large *	100,125	35,000	
i3.xlarge *	206,250	70,000	
i3.2xlarge	412,500	180,000	
i3.4xlarge	825,000	360,000	
i3.8xlarge	1.65 million	720,000	
i3.16xlarge	3.3 million	1.4 million	
i3.metal	3.3 million	1.4 million	
i3en.large *	42,500	32,500	
i3en.xlarge *	85,000	65,000	
i3en.2xlarge *	170,000	130,000	
i3en.3xlarge	250,000	200,000	
i3en.6xlarge	500,000	400,000	
i3en.12xlarge	1 million	800,000	
i3en.24xlarge	2 million	1.6 million	
i3en.metal	2 million	1.6 million	

© Stephane Maarek

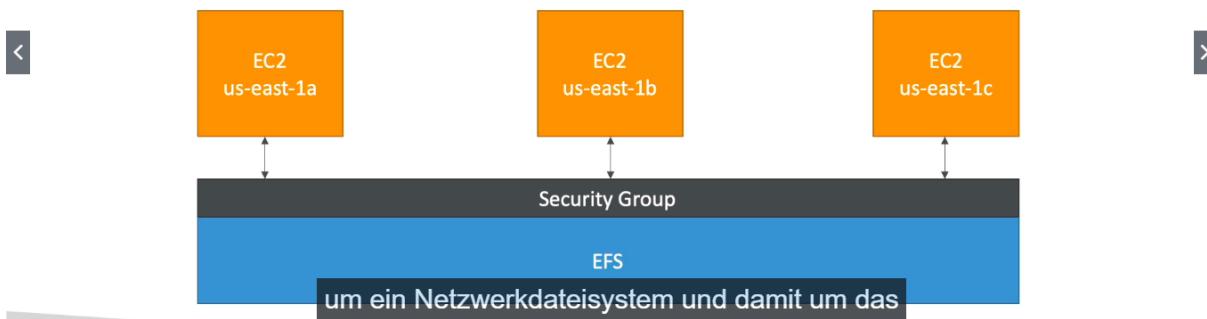
noch einmal gesagt, aber ich sage es Ihnen

↓↓↓↓↓

50. EFS Overview

EFS – Elastic File System

- Managed NFS (network file system) that can be mounted on many EC2
- EFS works with EC2 instances in multi-AZ
- Highly available, scalable, expensive (3x gp2), pay per use



um ein Netzwerkdateisystem und damit um das

© Stephane Maarek 1:33 / 4:38

↓↓↓↓↓

EFS – Elastic File System

- Use cases: content management, web serving, data sharing, Wordpress
 - Uses NFSv4.1 protocol
 - Uses security group to control access to EFS
 - Compatible with Linux based AMI (not Windows)
 - Encryption at rest using KMS
-
- POSIX file system (~Linux) that has a standard file API
 - File system scales automatically, pay-per-use, no capacity planning!

© Stephane Maarek

keine Kapazitätsplanung, sodass die Verwendung sehr einfach ist.

by tommy

EFS – Performance & Storage Classes

- EFS Scale
 - 1000s of concurrent NFS clients, 10 GB+ /s throughput
 - Grow to Petabyte-scale network file system, automatically
- Performance mode (set at EFS creation time)
 - General purpose (default): latency-sensitive use cases (web server, CMS, etc...)
 - Max I/O – higher latency, throughput, highly parallel (big data, media processing)
- Storage Tiers (lifecycle management feature – move file after N days)
 - Standard: for frequently accessed files
 - Infrequent access (EFS-IA): cost to retrieve files, lower price to store

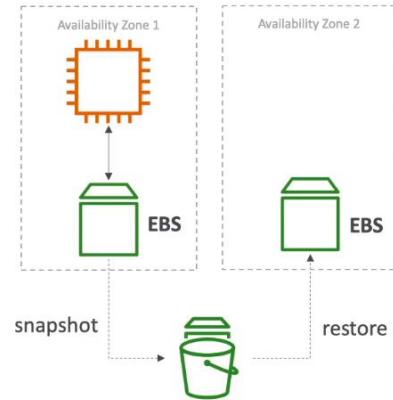
© Stephane Maarek

auf die nur selten zugegriffen wird

by tommy

EBS vs EFS – Elastic Block Storage

- EBS volumes...
 - can be attached to only one instance at a time
 - are locked at the Availability Zone (AZ) level
 - gp2: IO increases if the disk size increases
 - io1: can increase IO independently
- To migrate an EBS volume across AZ
 - Take a snapshot
 - Restore the snapshot to another AZ
 - EBS backups use IO and you shouldn't run them while your application is handling a lot of traffic
- Root EBS Volumes of instances get terminated by default if the EC2 instance gets terminated. (you can disable that)



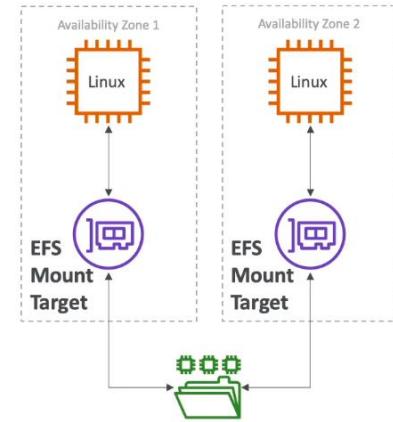
Schließlich werden die Root-EBS-Volumes Ihrer Instanzen standardmäßig beendet,

© Stephane Maarek

by Sunny

EBS vs EFS – Elastic File System

- Mounting 100s of instances across AZ
- EFS share website files (WordPress)
- Only for Linux Instances (POSIX)
- EFS has a higher price point than EBS
- Can leverage EFS-IA for cost savings
- Remember: EFS vs EBS vs Instance Store



das über mehrere Instanzen hinweg bereitgestellt werden kann.

© Stephane Maarek

by Sunny

AWS RDS Overview



- RDS stands for Relational Database Service
- It's a managed DB service for DB use SQL as a query language.
- It allows you to create databases in the cloud that are managed by AWS
 - Postgres
 - MySQL
 - MariaDB
 - Oracle
 - Microsoft SQL Server
 - Aurora (AWS Proprietary database)

© Stephane Maarek

Aber die ersten fünf, Postgres, MySQL, MariaDB,

by LinkedIn

Advantage over using RDS versus deploying DB on EC2

- RDS is a managed service:
 - Automated provisioning, OS patching
 - Continuous backups and restore to specific timestamp (Point in Time Restore)!
 - Monitoring dashboards
 - Read replicas for improved read performance
 - Multi AZ setup for DR (Disaster Recovery)
 - Maintenance windows for upgrades
 - Scaling capability (vertical and horizontal)
 - Storage backed by EBS (gp2 or io1)
- BUT you can't SSH into your instances

© Stephane Maarek

nicht in die Instanzen, die RDS-Instanzen, SSH können.

by LinkedIn

RDS Backups

- Backups are automatically enabled in RDS
- Automated backups:
 - Daily full backup of the database (during the maintenance window)
 - Transaction logs are backed-up by RDS every 5 minutes
 - => ability to restore to any point in time (from oldest backup to 5 minutes ago)
 - 7 days retention (can be increased to 35 days)
- DB Snapshots:
 - Manually triggered by the user
 - Retention of backup for as long as you want

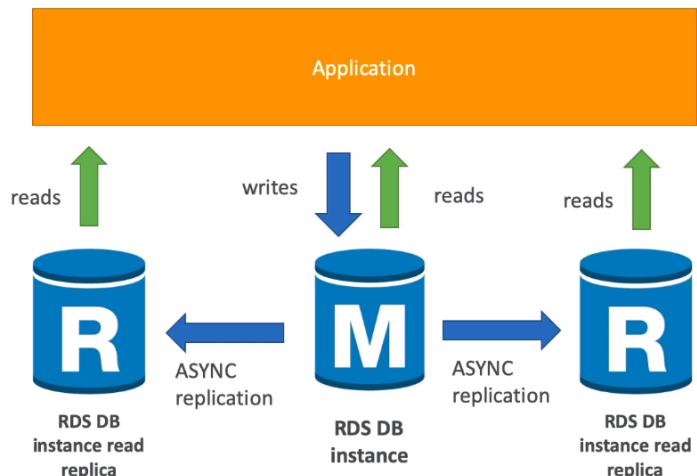
zu einem bestimmten Zeitpunkt sechs Monate lang beibehalten möchten.

© Stephane Maarek

by sunny

RDS Read Replicas for read scalability

- Up to 5 Read Replicas
- Within AZ, Cross AZ or Cross Region
- Replication is ASYNC, so reads are eventually consistent
- Replicas can be promoted to their own DB
- Applications must update the connection string to leverage read replicas



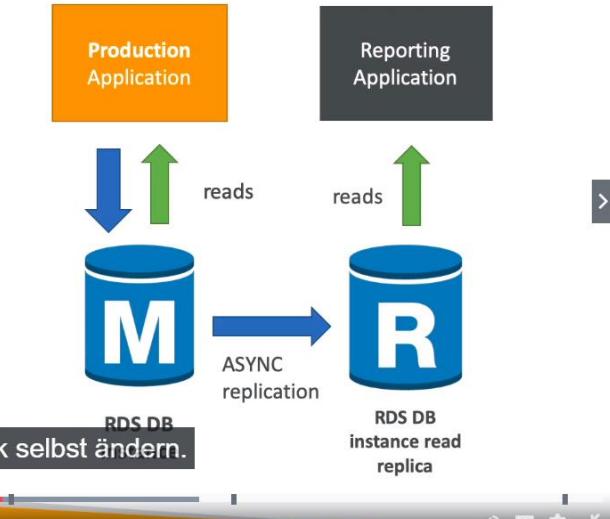
© Stephane Maarek

by sunny

55. RDS Read Replicas vs Multi AZ

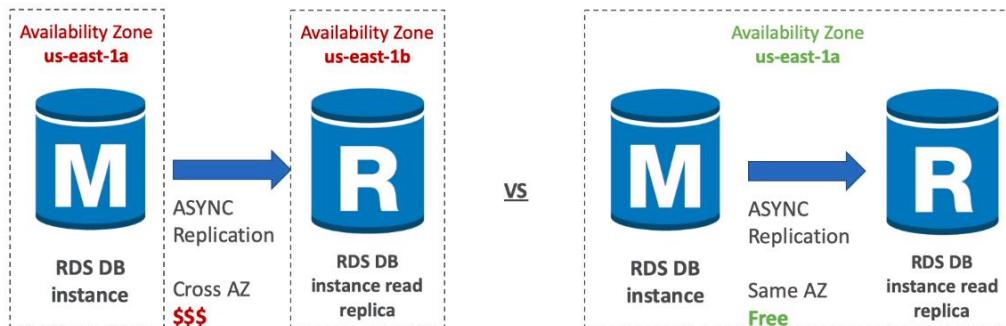
RDS Read Replicas – Use Cases

- You have a production database that is taking on normal load
- You want to run a reporting application to run some analytics
- You create a Read Replica to run the new workload there
- The production application is unaffected
- Read replicas are used for SELECT (=read) only kind of statements (not INSERT, UPDATE, DELETE)



RDS Read Replicas – Network Cost

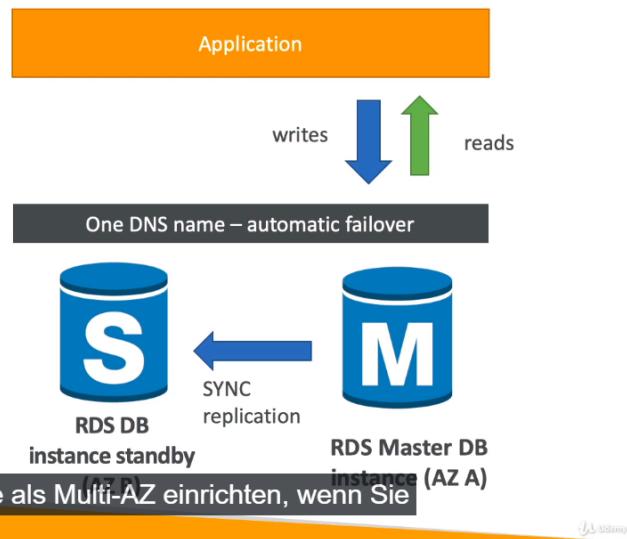
- In AWS there's a network cost when data goes from one AZ to another
- To reduce the cost, you can have your Read Replicas in the same AZ



Ich denke, das macht

RDS Multi AZ (Disaster Recovery)

- SYNC replication
- One DNS name – automatic app failover to standby
- Increase availability
- Failover in case of loss of AZ, loss of network, instance or storage failure
- No manual intervention in apps
- Not used for scaling
- Note: The Read Replicas be setup as Multi AZ for Disaster Recovery (DR)



© Stephane Maarek

Sie können Ihre Lesereplikate als Multi-AZ einrichten, wenn Sie

JB Savary

RDS Security - Encryption

- At rest encryption
 - Possibility to encrypt the master & read replicas with AWS KMS - AES-256 encryption
 - Encryption has to be defined at launch time
 - If the master is not encrypted, the read replicas cannot be encrypted
 - Transparent Data Encryption (TDE) available for Oracle and SQL Server
- In-flight encryption
 - SSL certificates to encrypt data to RDS in flight
 - Provide SSL options with trust certificate when connecting to database
 - To enforce SSL:
 - PostgreSQL: rds.force_ssl=1 in the AWS RDS Console (Parameter Groups)
 - MySQL: Within the DB:
GRANT USAGE ON *.* TO 'mysqluser'@'%' REQUIRE SSL;

© Stephane Maarek

Parametergruppe und MySQL wird ein SQL-Befehl in der Datenbank sein.

JB Savary

RDS Encryption Operations

- Encrypting RDS backups
 - Snapshots of un-encrypted RDS databases are un-encrypted
 - Snapshots of encrypted RDS databases are encrypted
 - Can copy a snapshot into an encrypted one
- To encrypt an un-encrypted RDS database:
 - Create a snapshot of the un-encrypted database
 - Copy the snapshot and enable encryption for the snapshot
 - Restore the database from the encrypted snapshot
 - Migrate applications to the new database, and delete the old database

einmal sehen und wenn es in der Prüfung auftaucht, wissen Sie davon.

© Stephane Maarek

by 

57. RDS Encryption + Security

RDS Security – Network & IAM

- Network Security
 - RDS databases are usually deployed within a private subnet, not in a public one
 - RDS security works by leveraging security groups (the same concept as for EC2 instances) – it controls which IP / security group can communicate with RDS
- Access Management
 - IAM policies help control who can manage AWS RDS (through the RDS API)
 - Traditional Username and Password can be used to login into the database
 - IAM-based authentication can be used to login into RDS MySQL & PostgreSQL

werden, können Sie IAM-basiert verwenden Authentifizierung, okay?

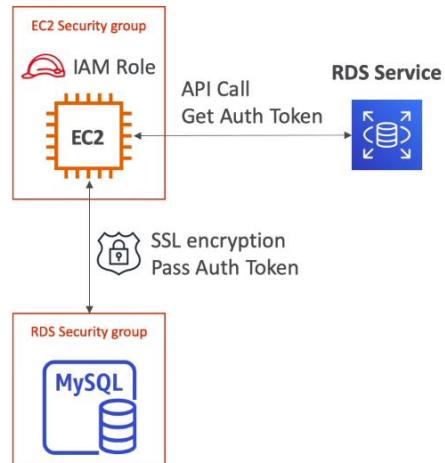
© Stephane Maarek

by 

3:54 / 6:41

RDS - IAM Authentication

- IAM database authentication works with MySQL and PostgreSQL
- You don't need a password, just an authentication token obtained through IAM & RDS API calls
- Auth token has a lifetime of 15 minutes
- Benefits:
 - Network in/out must be encrypted using SSL
 - IAM to centrally manage users instead of DB
 - Can leverage IAM Roles and EC2 Instance profiles for easy integration



© Stephane Maarek

sehr bald sehen, was IAM-Rollen- und EC2-Instanzprofile sind.

15. Summary

57. RDS Encryption + Security

RDS Security – Summary

- Encryption at rest:
 - Is done only when you first create the DB instance
 - or: unencrypted DB => snapshot => copy snapshot as encrypted => create DB from snapshot
- Your responsibility:
 - Check the ports / IP / security group inbound rules in DB's SG
 - In-database user creation and permissions or manage through IAM
 - Creating a database with or without public access
 - Ensure parameter groups or DB is configured to only allow SSL connections
- AWS responsibility:
 - No SSH access
 - No manual DB patching
 - No manual OS patching
 - No way to audit the underlying instance

© Stephane Maarek 5.29 / 601

Das RDS ist also ein Service, der Ihnen angeboten

15. Summary

Amazon Aurora

- Aurora is a proprietary technology from AWS (not open sourced)
- Postgres and MySQL are both supported as Aurora DB (that means your drivers will work as if Aurora was a Postgres or MySQL database)
- Aurora is “AWS cloud optimized” and claims 5x performance improvement over MySQL on RDS, over 3x the performance of Postgres on RDS
- Aurora storage automatically grows in increments of 10GB, up to 64 TB.
- Aurora can have 15 replicas while MySQL has 5, and the replication process is faster (sub 10 ms replica lag)
- Failover in Aurora is instantaneous. It’s HA native.
- Aurora costs more than RDS (20% more) – but is more efficient

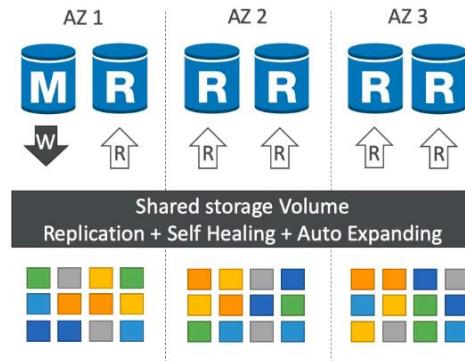
© Stephane Maarek

bei RDS, etwa 20% mehr, ist es jetzt so viel

by sunny

Aurora High Availability and Read Scaling

- 6 copies of your data across 3 AZ:
 - 4 copies out of 6 needed for writes
 - 3 copies out of 6 need for reads
 - Self healing with peer-to-peer replication
 - Storage is striped across 100s of volumes
- One Aurora Instance takes writes (master)
- Automated failover for master in less than 30 seconds
- Master + up to 15 Aurora Read Replicas serve reads
- Support for Cross Region Replication

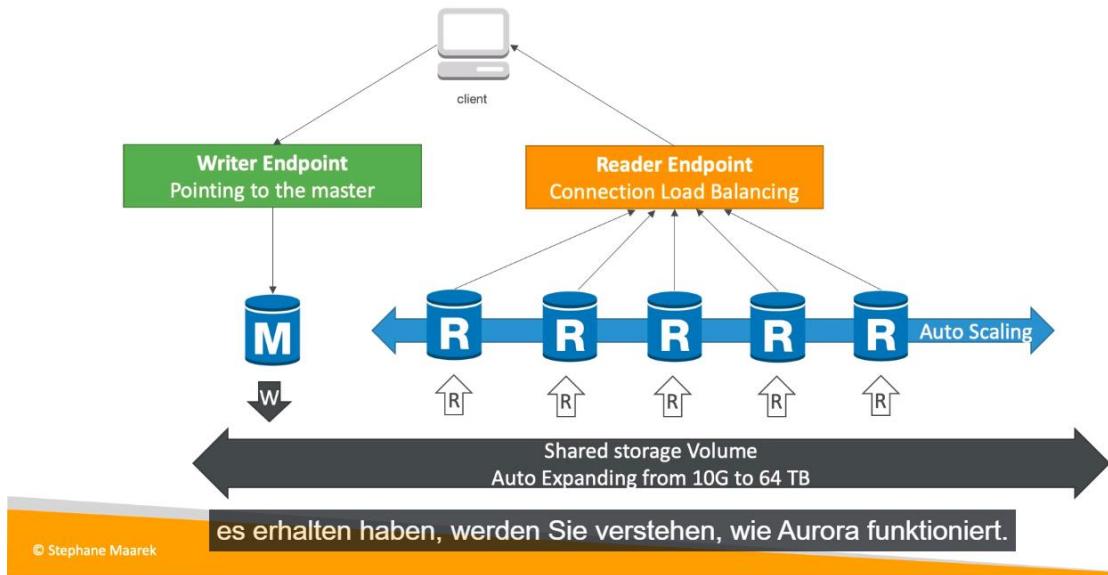


© Stephane Maarek

Schauen wir uns nun an, wie Aurora als Cluster ist.

by sunny

Aurora DB Cluster



Features of Aurora

- Automatic fail-over
- Backup and Recovery
- Isolation and security
- Industry compliance
- Push-button scaling
- Automated Patching with Zero Downtime
- Advanced Monitoring
- Routine Maintenance
- Backtrack: restore data at any point of time without using backups



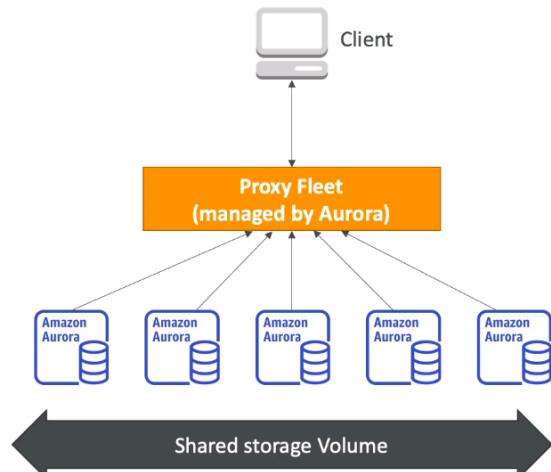
Aurora Security

- Similar to RDS because uses the same engines
- Encryption at rest using KMS
- Automated backups, snapshots and replicas are also encrypted
- Encryption in flight using SSL (same process as MySQL or Postgres)
- Possibility to authenticate using IAM token (same method as RDS)
- You are responsible for protecting the instance with security groups
- You can't SSH

© Stephane Maarek die Aurora-Sicherheit genau der RDS-Sicherheit. 

Aurora Serverless

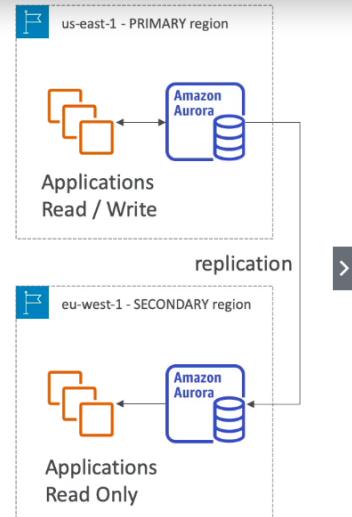
- Automated database instantiation and auto-scaling based on actual usage
- Good for infrequent, intermittent or unpredictable workloads
- No capacity planning needed
- Pay per second, can be more cost-effective



© Stephane Maarek und diese wird je nach Bedarf skaliert. 

Global Aurora

- Aurora Cross Region Read Replicas:
 - Useful for disaster recovery
 - Simple to put in place
- Aurora Global Database (recommended):
 - 1 Primary Region (read / write)
 - Up to 5 secondary (read-only) regions, replication lag is less than 1 second
 - Up to 16 Read Replicas per secondary region
 - Helps for decreasing latency
 - Promoting another region (for disaster recovery) has an RTO of < 1 minute



Thema, aber es ist wirklich

AWS ElastiCache Overview



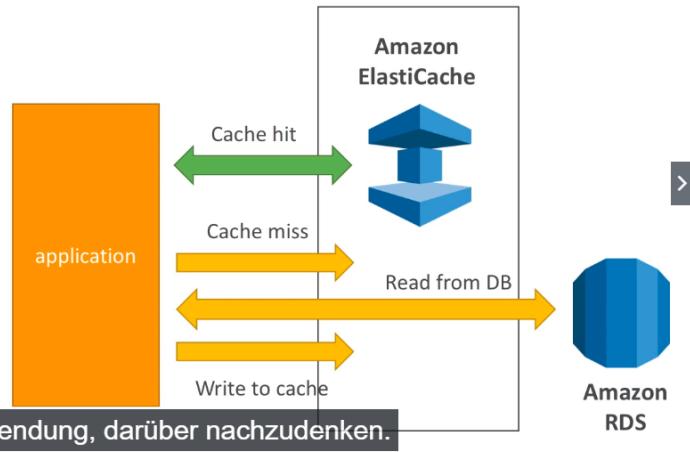
- The same way RDS is to get managed Relational Databases...
- ElastiCache is to get managed Redis or Memcached
- Caches are in-memory databases with really high performance, low latency
- Helps reduce load off of databases for read intensive workloads
- Helps make your application stateless
- Write Scaling using sharding
- Read Scaling using Read Replicas
- Multi AZ with Failover Capability
- AWS takes care of OS maintenance / patching, optimizations, setup, configuration, monitoring, failure recovery and backups

Im Grunde sieht es RDS sehr ähnlich und es gibt

ElastiCache

Solution Architecture - DB Cache

- Applications queries ElastiCache, if not available, get from RDS and store in ElastiCache.
- Helps relieve load in RDS
- Cache must have an invalidation strategy to make sure only the most current data is used in there.

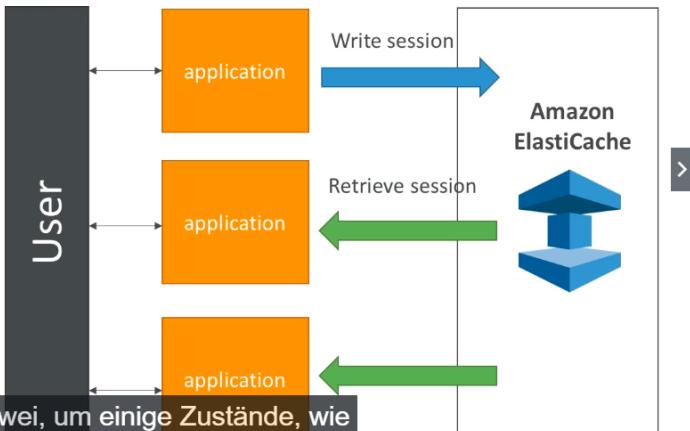


ElastiCache

Solution Architecture – User Session Store

- User logs into any of the application
- The application writes the session data into ElastiCache
- The user hits another instance of our application
- The instance retrieves the data and the user is already logged in

die Nummer zwei, um einige Zustände, wie



ElastiCache – Redis vs Memcached

REDIS

- Multi AZ with Auto-Failover
- Read Replicas to scale reads and have high availability
- Data Durability using AOF persistence
- Backup and restore features

MEMCACHED

- Multi-node for partitioning of data (sharding)
- Non persistent
- No backup and restore
- Multi-threaded architecture



geben, keine Persistenz, Multi -threaded Architektur und so weiter.

© Stephane Maarek

b1 salamy

Caching Implementation Considerations

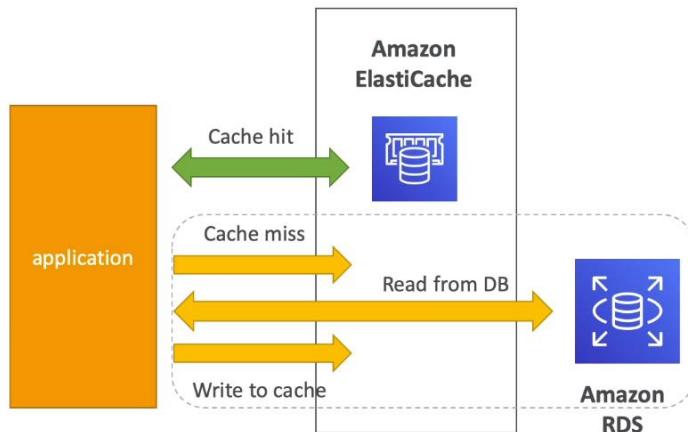
- Read more at: <https://aws.amazon.com/caching/implementation-considerations/>
- Is it safe to cache data? Data may be out of date, eventually consistent
- Is caching effective for that data?
 - Pattern: data changing slowly, few keys are frequently needed
 - Anti patterns: data changing rapidly, all large key space frequently needed
- Is data structured well for caching?
 - example: key value caching, or caching of aggregations results
- Which caching design pattern is the most appropriate?

Caching-Designmuster für uns am besten geeignet ist.

© Stephane Maarek

b1 salamy

Lazy Loading / Cache-Aside / Lazy Population



- Pros
 - Only requested data is cached (the cache isn't filled up with unused data)
 - Node failures are not fatal (just increased latency to warm the cache)
- Cons
 - Cache miss penalty that results in 3 round trips, noticeable delay for that request
 - Stale data: data can be updated in the database and outdated in the cache

© Stephane Maarek Wenn Ihre Daten in RDS aktualisiert werden,

bbb sunny

Lazy Loading / Cache-Aside / Lazy Population Python Pseudocode

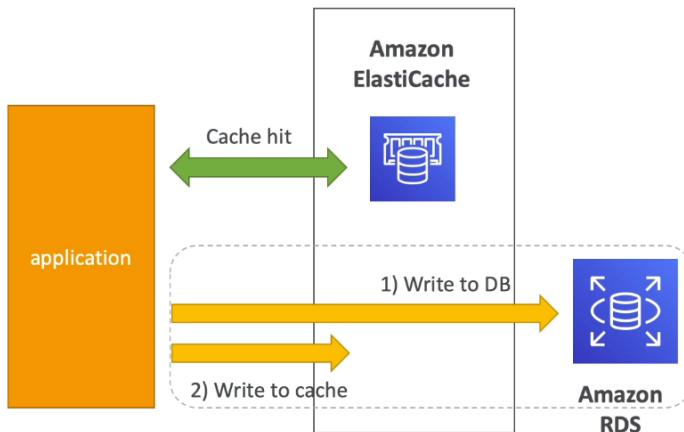
```
1  # Python
2
3  def get_user(user_id):
4      # Check the cache
5      record = cache.get(user_id)
6
7      if record is None:
8          # Run a DB query
9          record = db.query("select * from users where id = ?", user_id)
10         # Populate the cache
11         cache.set(user_id, record)
12         return record
13     else:
14         return record
15
16     # App code
17     user = get_user(17)
```

wenig von der Prüfung, aber für ElastiCache ist es notwendig,
okay.

© Stephane Maarek

bbb sunny

Write Through – Add or Update cache when database is updated



- Pros:

- Data in cache is never stale, reads are quick
- Write penalty vs Read penalty (each write requires 2 calls)

- Cons:

- Missing Data until it is added / updated in the DB. Mitigation is to implement Lazy Loading strategy as well
- Cache churn – a lot of the data will never be read

© Stephane Maarek

und viele Daten in ElastiCache vorhanden sind, aber

↓↓↓↓↓

62. ElastiCache Strategies

Write-Through Python Pseudocode

```
1  # Python
2
3  def save_user(user_id, values):
4
5      # Save to DB
6
7      record = db.query("update users ... where id = ?", user_id, values)
8
9      # Push into cache
10
11     cache.set(user_id, record)
12
13     return record
14
15 # App code
16
17 user = save_user(17, {"name": "Nate Dogg"})
```

die andere eine Funktion namens get_user

© Stephane Maarek 8:47 / 11:37

CC BY NC SA

Cache Evictions and Time-to-live (TTL)

- Cache eviction can occur in three ways:
 - You delete the item explicitly in the cache
 - Item is evicted because the memory is full and it's not recently used (LRU)
 - You set an item time-to-live (or TTL)
- TTL are helpful for any kind of data:
 - Leaderboards
 - Comments
 - Activity streams
- TTL can range from few seconds to hours or days
- If too many evictions happen due to memory, you should scale up or out

The screenshot shows a presentation slide with the following elements:

- Header:** "Verkleinern zu aktualisieren." (Minimize to update.)
- Section:** "62. ElastiCache Strategies"
- Title:** "Final words of wisdom"
- Content:**
 - Lazy Loading / Cache aside is easy to implement and works for many situations as a foundation, especially on the read side
 - Write-through is usually combined with Lazy Loading as targeted for the queries or workloads that benefit from this optimization
 - Setting a TTL is usually not a bad idea, except when you're using Write-through. Set it to a sensible value for your application
 - Only cache the data that makes sense (user profiles, blogs, etc...)
 - Quote: *There are only two hard things in Computer Science: cache invalidation and naming things*
- Footer:** "das Caching wirklich sehr, sehr schwierig ist."
- Navigation:** Left and right arrows for navigation.
- Player controls:** Standard video player controls (play/pause, volume, etc.) and a progress bar showing 11:22 / 11:37.

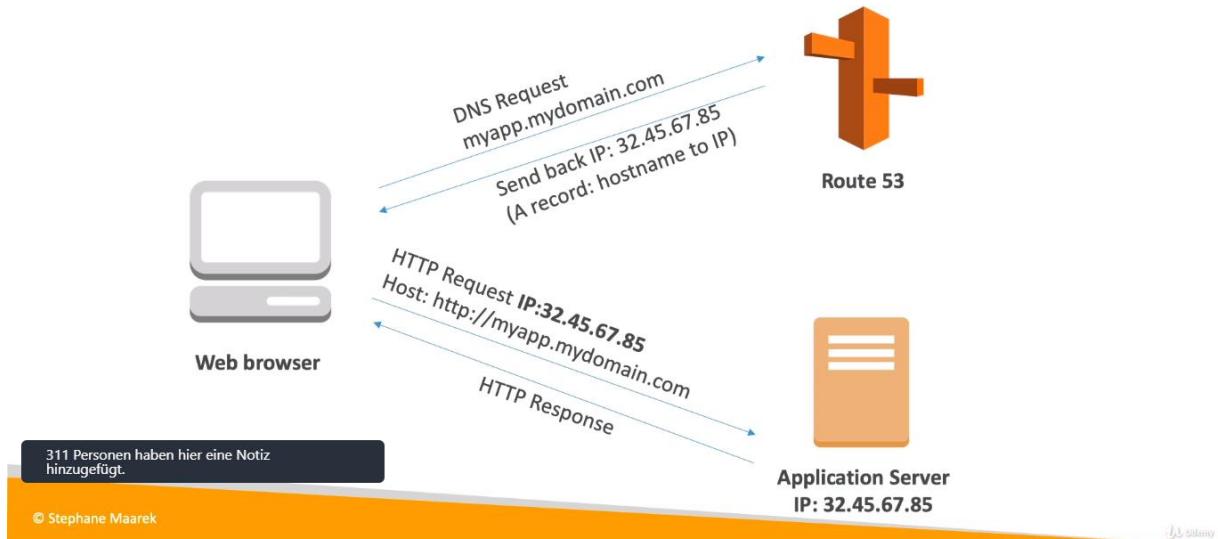
AWS Route 53 Overview



- Route53 is a Managed DNS (Domain Name System)
- DNS is a collection of rules and records which helps clients understand how to reach a server through its domain name.
- In AWS, the most common records are:
 - A: hostname to IPv4
 - AAAA: hostname to IPv6
 - CNAME: hostname to hostname
 - Alias: hostname to AWS resource.



Route 53 – Diagram for A Record



63. Route 53 Overview

AWS Route 53 Overview

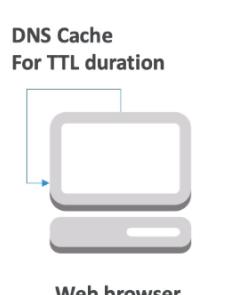
- Route53 can use:
 - public domain names you own (or buy)
[application1.mypublicdomain.com](#)
 - private domain names that can be resolved by your instances in your VPCs.
[application1.company.internal](#)
- Route53 has advanced features such as:
 - Load balancing (through DNS – also called client load balancing)
 - Health checks (although limited...)
 - Routing policy: simple, failover, geolocation, latency, weighted, multi value
- You pay \$0.50 per month per hosted zone

311 Personen haben hier eine Notiz hinzugefügt.

o, nur damit Sie wissen, wenn Sie mit mir



DNS Records TTL (Time to Live)



DNS Request
http://myapp.mydomain.com

Send back IP: 32.45.67.85
(A record: URL to IP)
+ TTL : 300 s

DNS Request
http://myapp.mydomain.com

Send back IP: 195.23.45.22
(A record: URL to IP)
+ TTL : 300 s



Route 53

- High TTL: (e.g. 24hr)
 - Less traffic on DNS
 - Possibly outdated records
- Low TTL: (e.g 60 s)
 - More traffic on DNS
 - Records are outdated for less time
 - Easy to change records
- TTL is mandatory for each DNS record

© Stephane Maarek

Für jeden DNS-Eintrag muss eine TTL angegeben werden.

b1 summary

CNAME vs Alias

- AWS Resources (Load Balancer, CloudFront...) expose an AWS hostname: lb1-1234.us-east-2.elb.amazonaws.com and you want myapp.mydomain.com
- CNAME:
 - Points a hostname to any other hostname. (app.mydomain.com => blabla.anything.com)
 - ONLY FOR NON ROOT DOMAIN (ex: something.mydomain.com)
- Alias:
 - Points a hostname to an AWS Resource (app.mydomain.com => blabla.amazonaws.com)
 - Works for ROOT DOMAIN and NON ROOT DOMAIN (aka mydomain.com)
 - Free of charge
 - Native health check

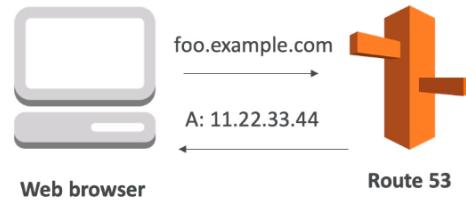
lassen Sie uns in die Hand gehen, um zu sehen, was ich damit meine.

© Stephane Maarek

by tommy

Simple Routing Policy

- Use when you need to redirect to a single resource
- You can't attach health checks to simple routing policy
- If multiple values are returned, a random one is chosen by the client



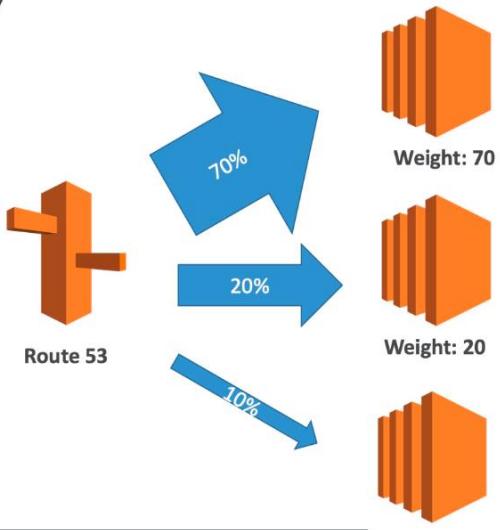
Schauen wir uns also an, wie dies in der Praxis funktioniert.

© Stephane Maarek

by tommy

Weighted Routing Policy

- Control the % of the requests that go to specific endpoint
- Helpful to test 1% of traffic on new app version for example
- Helpful to split traffic between two regions
- Can be associated with Health Checks



© Stephane Maarek

Schauen wir uns also an, wie dies in der Konsole funktioniert.

70. Routing Policy - Latency

Latency Routing Policy

• Redirect to the server that has the least latency close to us

• Super helpful when latency of users is a priority

• Latency is evaluated in terms of user to designated AWS Region

• Germany may be directed to the US (if that's the lowest latency)

Karte nach Australien umgeleitet werden

Wiedergabegeschwindigkeit: 1.5x

0:58 / 4:10

Health Checks

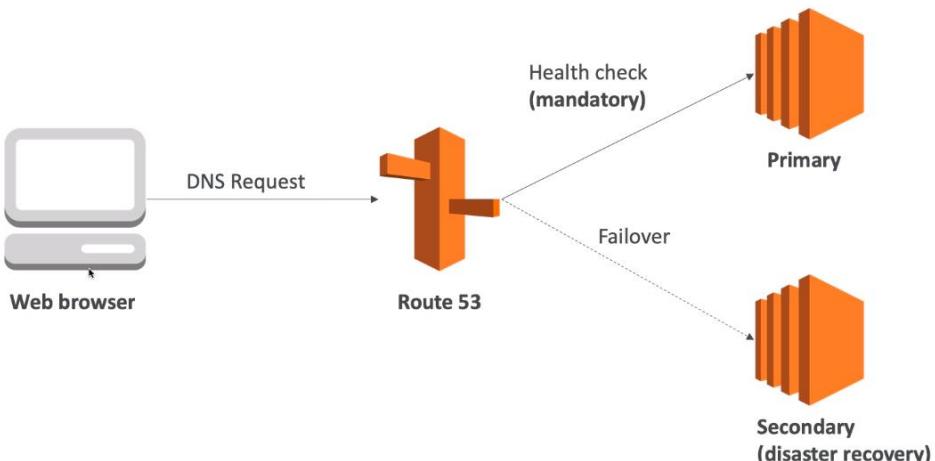
- Have X health checks failed => unhealthy (default 3)
- After X health checks passed => health (default 3)
- Default Health Check Interval: 30s (can set to 10s – higher cost)
- About 15 health checkers will check the endpoint health
- => one request every 2 seconds on average
- Can have HTTP,TCP and HTTPS health checks (no SSL verification)
- Possibility of integrating the health check with CloudWatch
- Health checks can be linked to Route53 DNS queries!

© Stephane Maarek

und das Verhalten von Route 53 grundlegend ändern.

by Sammy

Failover Routing Policy



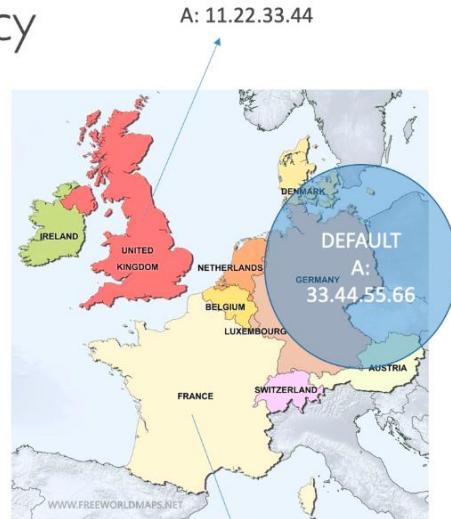
© Stephane Maarek

an den Webbrowser.

by Sammy

Geo Location Routing Policy

- Different from Latency based!
- This is routing based on user location
- Here we specify: traffic from the UK should go to this specific IP
- Should create a “default” policy (in case there’s no match on location)



© Stephane Maarek

Und so funktioniert die Geolokalisierung. Lassen

76. VPC Fundamentals - Section Introduction

VPC – Crash Course

- VPC is something you should know in depth for the AWS Certified Solutions Architect Associate & AWS Certified SysOps Administrator
- At the AWS Certified Developer Level, you should know about:
 - VPC, Subnets, Internet Gateways & NAT Gateways
 - Security Groups, Network ACL (NACL), VPC Flow Logs
 - VPC Peering, VPC Endpoints
 - Site to Site VPN & Direct Connect
- I will just give you an overview, less than 1 or 2 questions at your exam.
- Later in the course, I will be highlighting when VPC concepts are helpful

alles aus diesem Abschnitt erinnern.

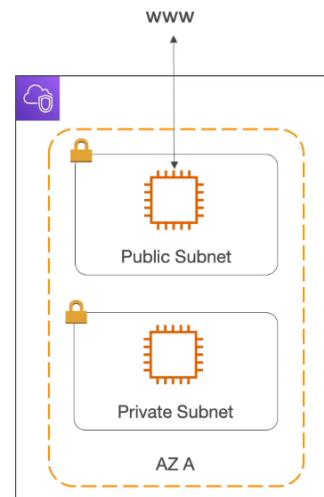
II 3 1x 1:02 / 1:23

Stephane Maarek

CC G

VPC & Subnets Primer

- VPC: private network to deploy your resources (regional resource)
- Subnets allow you to partition your network inside your VPC (Availability Zone resource)
- A public subnet is a subnet that is accessible from the internet
- A private subnet is a subnet that is not accessible from the internet
- To define access to the internet and between subnets, we use Route Tables.

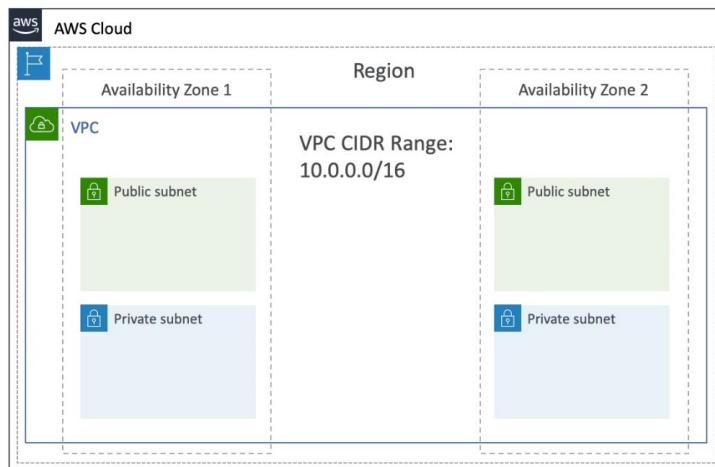


© Stephane Maarek

oder das Internet hat keinen Zugang dazu.

↓↓↓↓↓

VPC Diagram



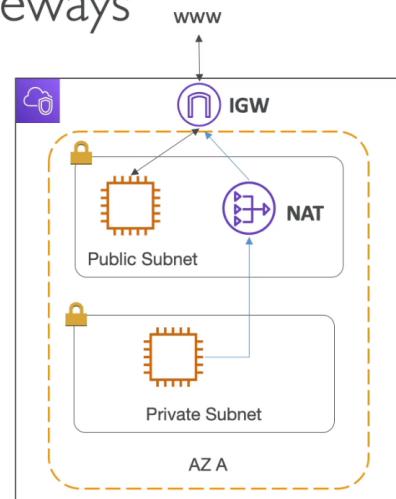
© Stephane Maarek

Es wird die Standard-VPC genannt.

↓↓↓↓↓

Internet Gateway & NAT Gateways

- Internet Gateways helps our VPC instances connect with the internet
 - Public Subnets have a route to the internet gateway.
- < >
- NAT Gateways (AWS-managed) & NAT Instances (self-managed) allow your instances in your Private Subnets to access the internet while remaining private



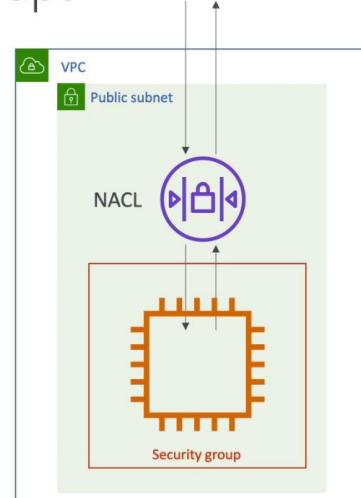
über die Nüsse bis zum Internet zugreifen können.

© Stephane Maarek

II 1x 4:51 / 5:23

Network ACL & Security Groups

- NACL (Network ACL)
 - A firewall which controls traffic from and to subnet
 - Can have ALLOW and DENY rules
 - Are attached at the Subnet level
 - Rules only include IP addresses
- Security Groups
 - A firewall that controls traffic to and from an ENI / an EC2 Instance
 - Can have only ALLOW rules
 - Rules include IP addresses and other security groups



die als Firewall fungiert.

© Stephane Maarek

II 1x 2:18 / 4:36

Network ACLs vs Security Groups

Security Group	Network ACL
Operates at the instance level	Operates at the subnet level
Supports allow rules only	Supports allow rules and deny rules
Is stateful: Return traffic is automatically allowed, regardless of any rules	Is stateless: Return traffic must be explicitly allowed by rules
We evaluate all rules before deciding whether to allow traffic	We process rules in number order when deciding whether to allow traffic
Applies to an instance only if someone specifies the security group when launching the instance, or associates the security group with the instance later on	Automatically applies to all instances in the subnets it's associated with (therefore, you don't have to rely on users to specify the security group)

https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Secur

Das ist also nur, wenn Sie neugierig sind.

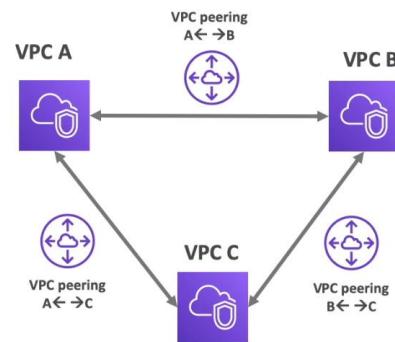
VPC Flow Logs



- Capture information about IP traffic going into your interfaces:
 - VPC Flow Logs
 - Subnet Flow Logs
 - Elastic Network Interface Flow Logs
- Helps to monitor & troubleshoot connectivity issues. Example:
 - Subnets to internet
 - Subnets to subnets
 - Internet to subnets
- Captures network information from AWS managed interfaces too: Elastic Load Balancers, ElastiCache, RDS, Aurora, etc...
- VPC Flow logs data can go to S3 / CloudWatch Logs

VPC Peering

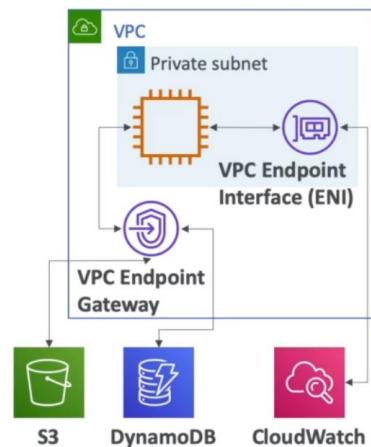
- Connect two VPC, privately using AWS' network
- Make them behave as if they were in the same network
- Must not have overlapping CIDR (IP address range)
- VPC Peering connection is **not** transitive (must be established for each VPC that need to communicate with one another)



das VPC-Peering. Wenn Sie also immer

VPC Endpoints

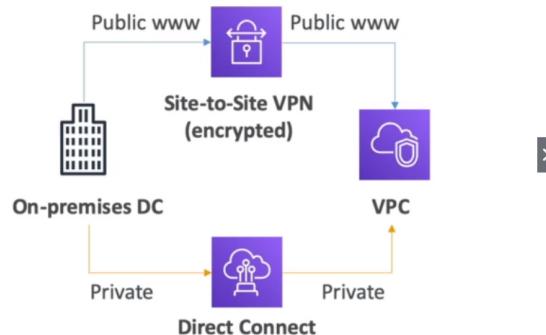
- Endpoints allow you to connect to AWS Services using a private network instead of the public www network
- This gives you enhanced security and lower latency to access AWS services
- VPC Endpoint Gateway: S3 & DynamoDB
- VPC Endpoint Interface: the rest
- Only used within your VPC



es ist, wenn Sie in der Prüfung aufgefordert werden, eine private

Site to Site VPN & Direct Connect

- Site to Site VPN
 - Connect an on-premises VPN to AWS
 - The connection is automatically encrypted
 - Goes over the public internet
- Direct Connect (DX)
 - Establish a physical connection between on-premises and AWS
 - The connection is private, secure and fast
 - Goes over a private network
 - Takes at least a month to establish
- Note: Site-to-site VPN and Direct Connect cannot access VPC endpoints



in Bezug auf externe VPC-Peering-VPC-Endpunkte wie

250 Personen haben hier eine Notiz hinzugefügt.

II 1x 5:41 / 5:50

80. VPC Cheat Sheet & Closing Comments

VPC Closing Comments

- VPC: Virtual Private Cloud
- Subnets: Tied to an AZ, network partition of the VPC
- Internet Gateway: at the VPC level, provide Internet Access
- NAT Gateway / Instances: give internet access to private subnets
- NACL: Stateless, subnet rules for inbound and outbound
- Security Groups: Stateful, operate at the EC2 instance level or ENI
- VPC Peering: Connect two VPC with non overlapping IP ranges, non transitive
- VPC Endpoints: Provide private access to AWS Services within VPC
- VPC Flow Logs: network traffic logs
- Site to Site VPN: VPN over public internet between on-premises DC and AWS
- Direct Connect: direct private connection to a AWS

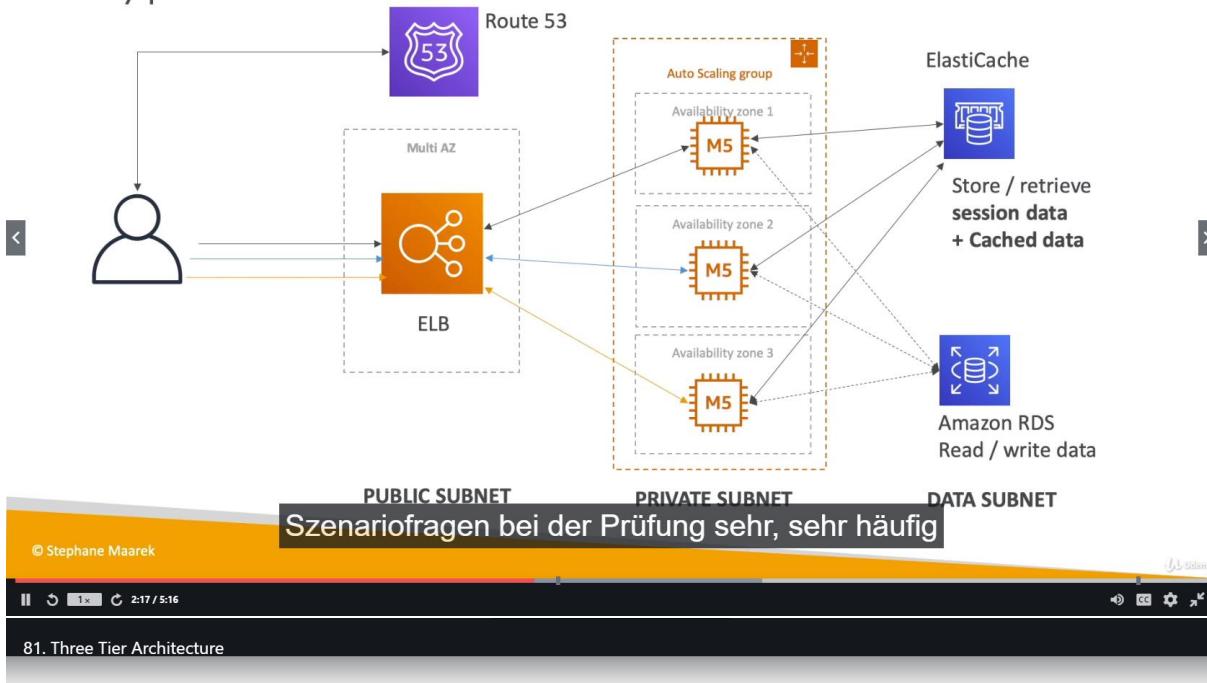
wenn Sie eine direkte private Verbindung zu AWS wünschen

© Stephane Maarek

II 1x 2:05 / 2:34

81. Three Tier Architecture

Typical 3 tier solution architecture



© Stephane Maarek

II ⏪ 1x ⏴ 2:17 / 5:16

Stephane Maarek

81. Three Tier Architecture

LAMP Stack on EC2

- Linux: OS for EC2 instances
 - Apache: Web Server that run on Linux (EC2)
 - MySQL: database on RDS
 - PHP: Application logic (running on EC2)
- < >
- Can add Redis / Memcached (ElastiCache) to include a caching tech
 - To store local application data & software: EBS drive (root)

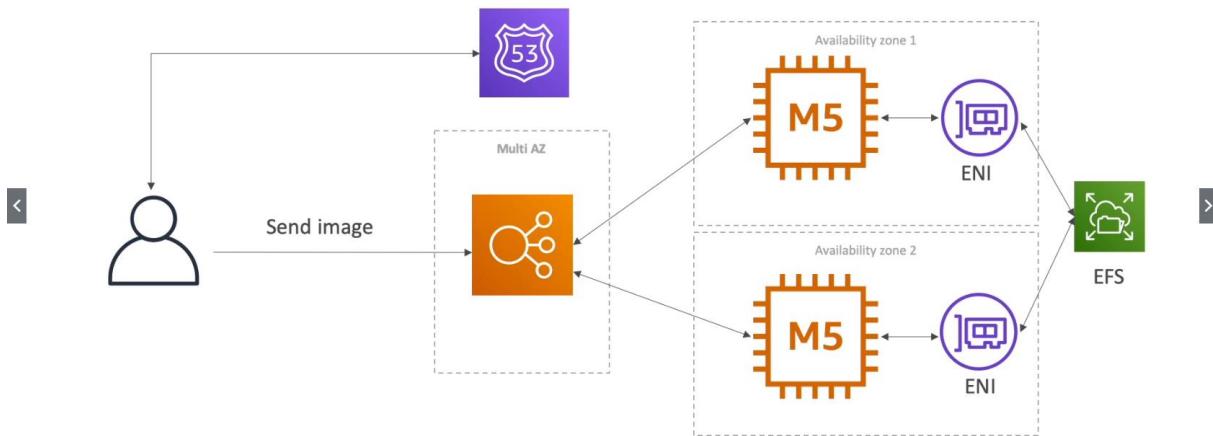
Das ist also wieder eine Idee einer Architektur in AWS.

© Stephane Maarek

II ⏪ 1x ⏴ 3:22 / 5:16

Stephane Maarek

Wordpress on AWS



Und so gibt es auf der

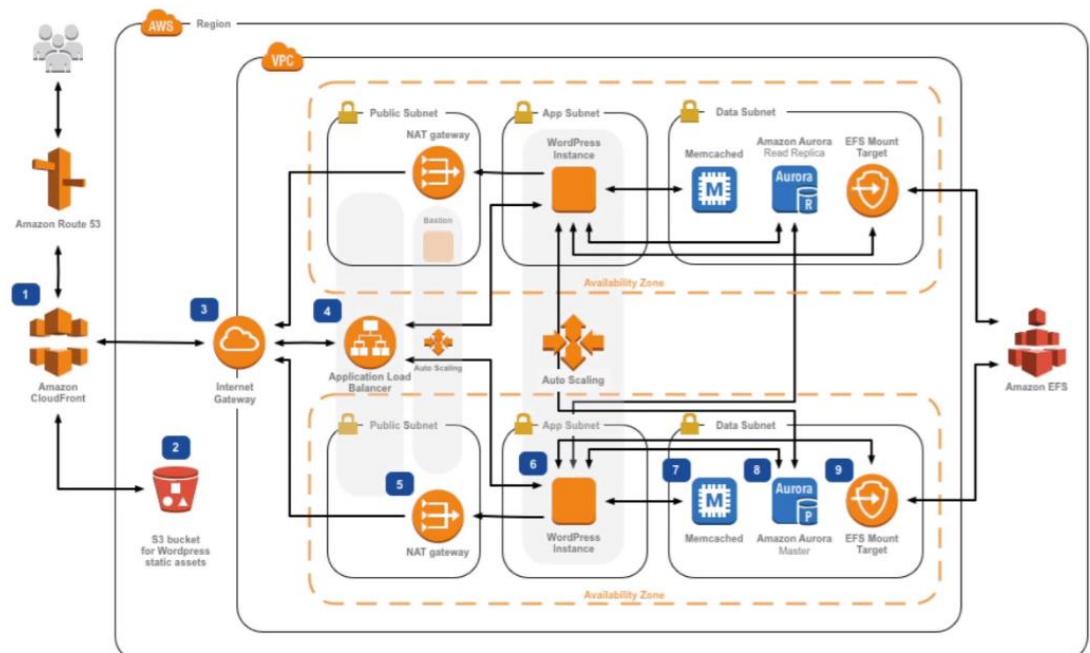
© Stephane Maarek

II ⏪ 1x ⏴ 4:24 / 5:16

Stephane Maarek

CC G

WordPress on AWS (more complicated)



Amazon S3 Overview - Buckets

- Amazon S3 allows people to store objects (files) in “buckets” (directories)
- Buckets must have a **globally unique name**
- Buckets are defined at the region level
- Naming convention
 - No uppercase
 - No underscore
 - 3-63 characters long
 - Not an IP
 - Must start with lowercase letter or number



© Stephane Maarek

sein. und es muss mit einem Kleinbuchstaben oder einer Zahl beginnen.

Amazon S3 Overview - Objects

- Objects (files) have a Key
- The **key** is the FULL path:
 - s3://my-bucket/my_file.txt
 - s3://my-bucket/my_folder1/another_folder/my_file.txt
- The key is composed of **prefix** + **object name**
 - s3://my-bucket/my_folder1/another_folder/my_file.txt
- There's no concept of “directories” within buckets (although the UI will trick you to think otherwise)
- Just keys with very long names that contain slashes (“/”)



632 Personen haben hier eine Notiz hinzugefügt.

mit sehr langen Namen, die Schrägstriche enthalten.

Amazon S3 Overview – Objects (continued)

- Object values are the content of the body:
 - Max Object Size is 5TB (5000GB)
 - If uploading more than 5GB, must use “multi-part upload”



- Metadata (list of text key / value pairs – system or user metadata)
- Tags (Unicode key / value pair – up to 10) – useful for security / lifecycle
- Version ID (if versioning is enabled)

The screenshot shows a video player interface with a yellow header bar. The header bar contains the text "in die Amazon S3-Konsole einsteigen und eine". Below the header, there is a dark content area with some small, illegible text.

Amazon S3 - Versioning



- You can version your files in Amazon S3
- It is enabled at the bucket level
- Same key overwrite will increment the “version”: 1, 2, 3....
- It is best practice to version your buckets
 - Protect against unintended deletes (ability to restore a version)
 - Easy roll back to previous version
- Notes:
 - Any file that is not versioned prior to enabling versioning will have version “null”
 - Suspending versioning does not delete the previous versions

The screenshot shows a video player interface with a yellow header bar. The header bar contains the text "die vorherigen Versionen nicht gelöscht. Es wird". Below the header, there is a dark content area with some small, illegible text.



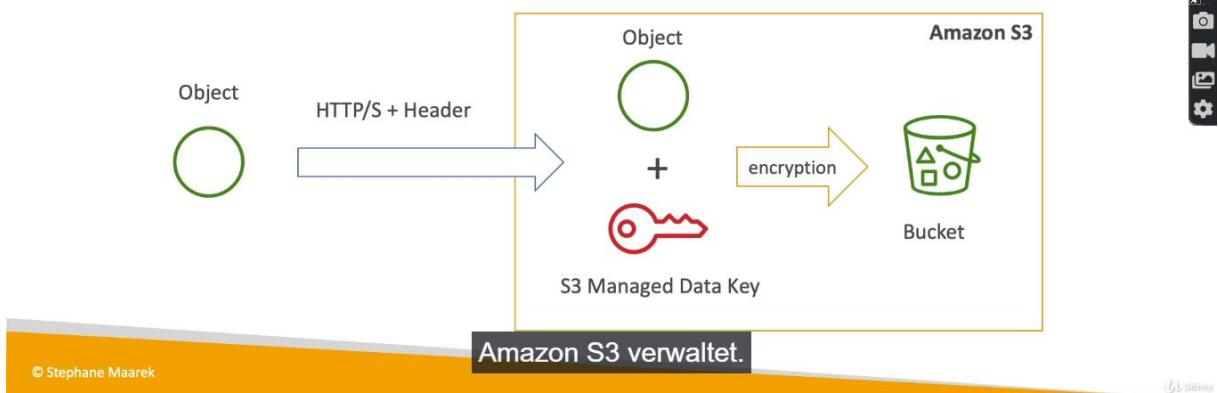
S3 Encryption for Objects

- There are 4 methods of encrypting objects in S3
 - SSE-S3: encrypts S3 objects using keys handled & managed by AWS
 - SSE-KMS: leverage AWS Key Management Service to manage encryption keys
 - SSE-C: when you want to manage your own encryption keys
 - Client Side Encryption
- It's important to understand which ones are adapted to which situation for the exam

© Stephane Maarek stellt, um die richtige Verschlüsselungsstufe basierend  

SSE-S3

- SSE-S3: encryption using keys handled & managed by Amazon S3
- Object is encrypted server side
- AES-256 encryption type
- Must set header: "x-amz-server-side-encryption": "AES256"

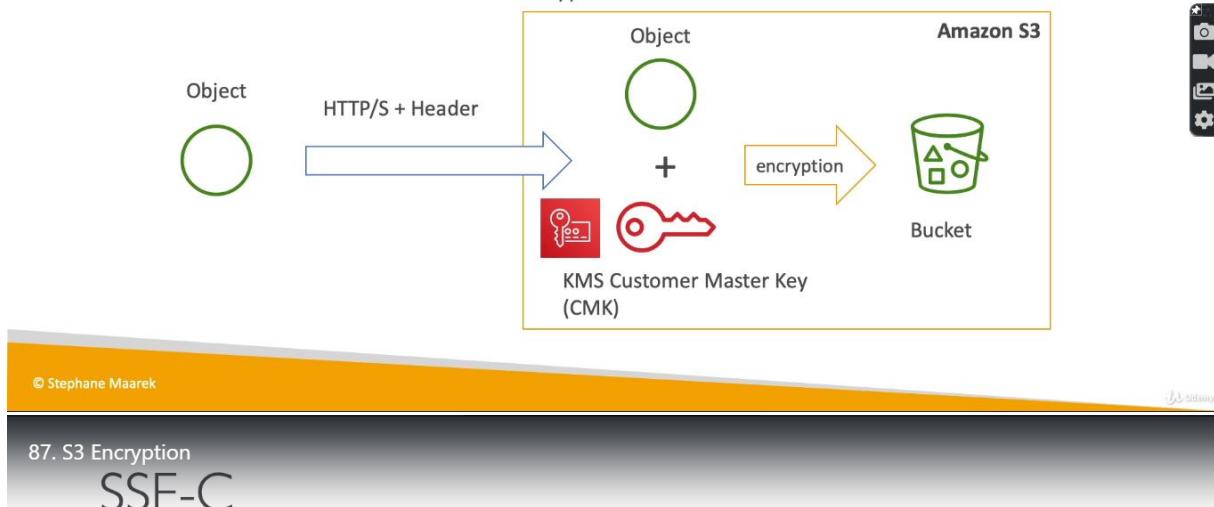


© Stephane Maarek  



SSE-KMS

- SSE-KMS: encryption using keys handled & managed by KMS
- KMS Advantages: user control + audit trail
- Object is encrypted server side
- Must set header: "x-amz-server-side-encryption": "aws:kms"



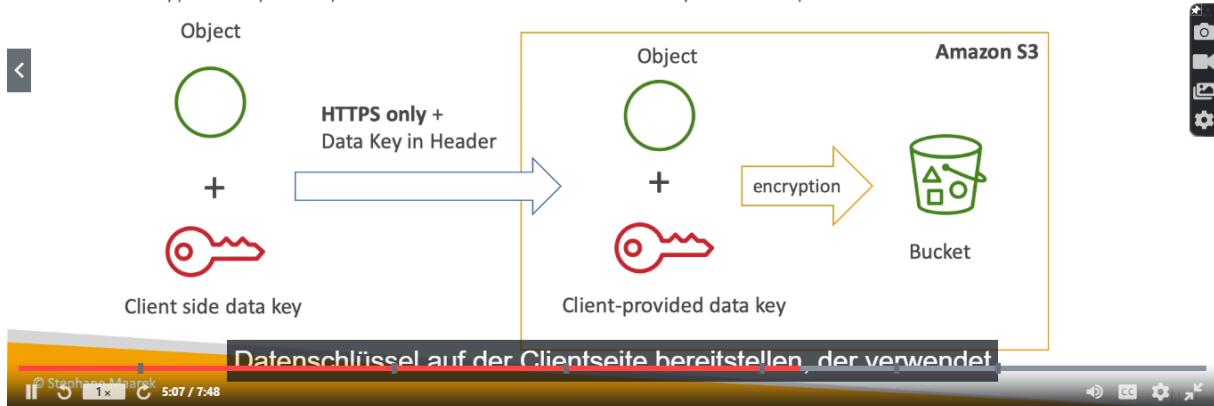
© Stephane Maarek

15:50 min

87. S3 Encryption

SSE-C

- SSE-C: server-side encryption using data keys fully managed by the customer outside of AWS
- Amazon S3 does not store the encryption key you provide
- HTTPS must be used
- Encryption key must provided in HTTP headers, for every HTTP request made



© Stephane Maarek

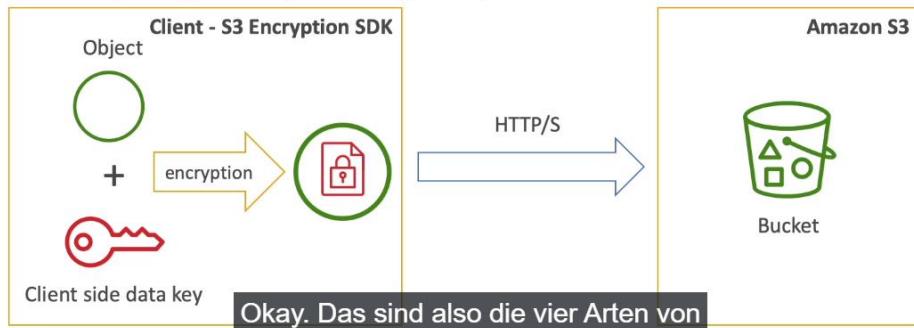
Datenschlüssel auf der Clientseite bereitstellen, der verwendet wird

15:50 min

87. S3 Encryption

Client Side Encryption

- Client library such as the Amazon S3 Encryption Client
- Clients must encrypt data themselves before sending to S3
- Clients must decrypt data themselves when retrieving from S3
- Customer fully manages the keys and encryption cycle



87. S3 Encryption

Encryption in transit (SSL/TLS)



- Amazon S3 exposes:
 - HTTP endpoint: non encrypted
 - HTTPS endpoint: encryption in flight
- You're free to use the endpoint you want, but HTTPS is recommended
- Most clients would use the HTTPS endpoint by default
- HTTPS is mandatory for SSE-C
- Encryption in flight is also called SSL / TLS





S3 Security

- User based
 - IAM policies - which API calls should be allowed for a specific user from IAM console
- Resource Based
 - Bucket Policies - bucket wide rules from the S3 console - allows cross account
 - Object Access Control List (ACL) – finer grain
 - Bucket Access Control List (ACL) – less common
- Note: an IAM principal can access an S3 object if
 - the user IAM permissions allow it OR the resource policy ALLOWS it
 - AND there's no explicit DENY

© Stephane Maarek Wenn Ihr Benutzer über IAM auf Ihren S3-Bucket zugreifen CC BY-SA

S3 Bucket Policies

- JSON based policies
 - Resources: buckets and objects
 - Actions: Set of API to Allow or Deny
 - Effect: Allow / Deny
 - Principal: The account or user to apply the policy to
- Use S3 bucket for policy to:
 - Grant public access to the bucket
 - Force objects to be encrypted at upload
 - Grant access to another account (Cross Account)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicRead",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::examplebucket/*"
      ]
    }
  ]
}
```



© Stephane Maarek auf ein anderes Konto mithilfe kontenübergreifender S3-Bucket-Richtlinien gewähren. CC BY-SA



Bucket settings for Block Public Access

- Block public access to buckets and objects granted through
 - new access control lists (ACLs)
 - any access control lists (ACLs)
 - new public bucket or access point policies
- Block public and cross-account access to buckets and objects through any public bucket or access point policies
- These settings were created to prevent company data leaks
- If you know your bucket should never be public, leave these on
- Can be set at the account level

Und es gibt eine Möglichkeit, diese auf Kontoebene festzulegen,
wie

© Stephane Maarek

by [Stephane Maarek](#)



S3 Security - Other

- Networking:
 - Supports VPC Endpoints (for instances in VPC without www internet)
- Logging and Audit:
 - S3 Access Logs can be stored in other S3 bucket
 - API calls can be logged in AWS CloudTrail
- User Security:
 - MFA Delete: MFA (multi factor authentication) can be required in versioned buckets to delete objects
 - Pre-Signed URLs: URLs that are valid only for a limited time (ex: premium video service for logged in users)

dieses Video gekauft hat.

© Stephane Maarek

by [Stephane Maarek](#)



S3 Websites

- S3 can host static websites and have them accessible on the www
- The website URL will be:
 - <bucket-name>.s3-website-<AWS-region>.amazonaws.com
 - OR
 - <bucket-name>.s3-website.<AWS-region>.amazonaws.com
- If you get a 403 (Forbidden) error, make sure the bucket policy allows public reads!

© Stephane Maarek Lassen Sie uns also unseren S3-Bucket als Website aktivieren. bb Sammy



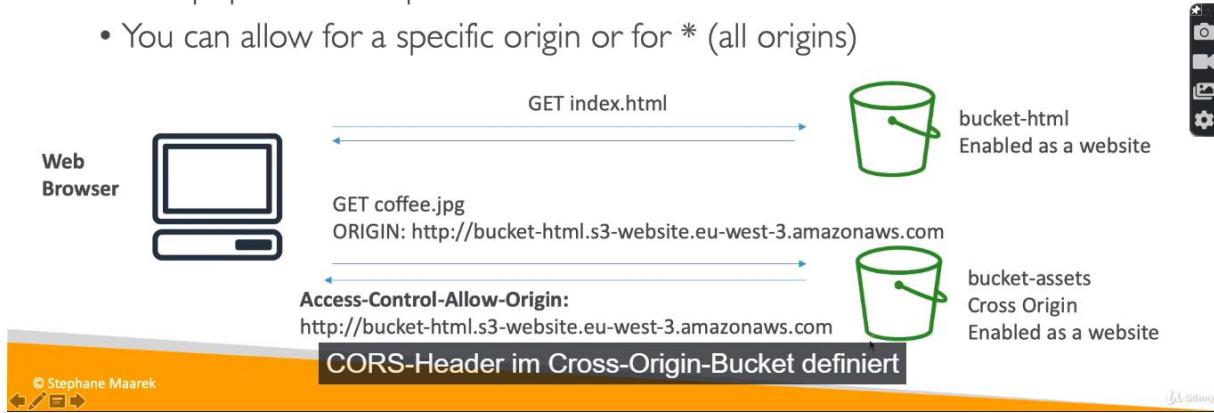
CORS - Explained

- An origin is a scheme (protocol), host (domain) and port
 - E.g.: <https://www.example.com> (implied port is 443 for HTTPS, 80 for HTTP)
- CORS means Cross-Origin Resource Sharing
- Web Browser based mechanism to allow requests to other origins while visiting the main origin
- Same origin: <http://example.com/app1> & <http://example.com/app2>
- Different origins: <http://www.example.com> & <http://otherexample.com>
- The requests won't be fulfilled unless the other origin allows for the requests, using CORS Headers (ex: Access-Control-Allow-Origin)

© Stephane Maarek Okay, das ist nur für die Theorie. bb Sammy

S3 CORS

- If a client does a cross-origin request on our S3 bucket, we need to enable the correct CORS headers
- It's a popular exam question
- You can allow for a specific origin or for * (all origins)



Amazon S3 - Consistency Model

- Read after write consistency for PUTS of new objects
 - As soon as a new object is written, we can retrieve it
ex: (PUT 200 => GET 200)
 - This is true, except if we did a GET before to see if the object existed
ex: (GET 404 => PUT 200 => GET 200) – eventually consistent
- Eventual Consistency for DELETES and PUTS of existing objects
 - If we read an object after updating, we might get the older version
ex: (PUT 200 => PUT 200 => GET 200 (might be older version))
 - If we delete an object, we might still be able to retrieve it for a short time
ex: (DELETE 200 => GET 200)
- Note: there's no way to request "strong consistency"

© Stephane Maarek es gibt keine API, um eine starke Konsistenz zu erzielen. b1 summary

92. S3 CORS

CORS – Diagram



99. AWS CLI Installation Troubleshooting

CLI Installation Troubleshooting

- If after installing the AWS CLI, you use it and you get the error

aws: command not found

- Then, on Linux, Mac and Windows

the aws executable is not in the PATH environment variable

- PATH allows your system to know where to find the “aws” executable

Das war's, nur ein sehr kurzer Vortrag.

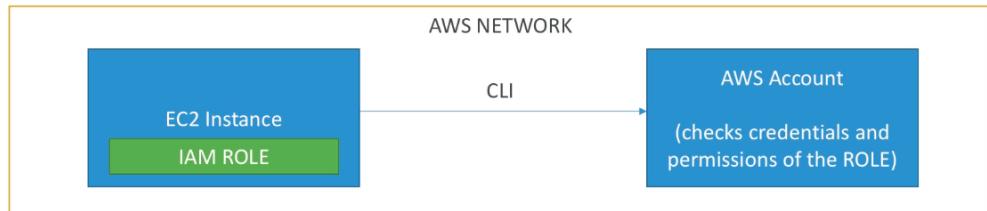
AWS CLI ON EC2... THE BAD WAY

- We could run `aws configure` on EC2 just like we did (and it'll work)
 - But... it's SUPER INSECURE
 - NEVER EVER EVER PUT YOUR PERSONAL CREDENTIALS ON AN EC2
 - Your PERSONAL credentials are PERSONAL and only belong on your PERSONAL computer
-
- If the EC2 is compromised, so is your personal account
 - If the EC2 is shared, other people may perform AWS actions while impersonating you
-
- For EC2, there's a better way... it's called AWS IAM Roles



AWS CLI ON EC2... THE RIGHT WAY

- IAM Roles can be attached to EC2 instances
- IAM Roles can come with a policy authorizing exactly what the EC2 instance should be able to do



- EC2 Instances can then use these profiles automatically without any additional configurations
- This is the best practice on AWS and you should 100% do this.

Verwenden Sie immer IAM-Rollen.



AWS CLI Dry Runs



- Sometimes, we'd just like to make sure we have the permissions...
- But not actually run the commands!

- Some AWS CLI commands (such as EC2) can become expensive if they succeed, say if we wanted to try to create an EC2 Instance
- Some AWS CLI commands (not all) contain a `--dry-run` option to simulate API calls

- Let's practice! **Berechtigungen verfügen, führen Sie den Befehl einfach nicht aus.**



AWS EC2 Instance Metadata



- AWS EC2 Instance Metadata is powerful but one of the least known features to developers
- It allows AWS EC2 instances to "learn about themselves" **without using an IAM Role for that purpose.**
- The URL is <http://169.254.169.254/latest/meta-data>
- You can retrieve the IAM Role name from the metadata, but you CANNOT retrieve the IAM Policy.
- Metadata = Info about the EC2 instance
- Userdata = launch script of the EC2 instance
- Let's practice and see what we can do with it!



```
[ec2-user@ip-172-31-3-136 ~]$ curl http://169.254.169.254/latest/
dynamic
meta-data
user-data[ec2-user@ip-172-31-3-136 ~]$ curl http://169.254.169.254/latest/meta-data/
ami-id
ami-launch-index
ami-manifest-path
block-device-mapping/
hostname
iam/
instance-action
instance-id
instance-type
local-hostname
local-ipv4
mac
metrics/
network/
placement/
profile
public-hostname
public-ipv4
public-keys/
reservation-id
security-groups
services/[ec2-user@ip-172-31-3-136 ~]$ curl http://169.254.169.254/latest/meta-data/instance-id
i-05adcce6933809eda[ec2-user@ip-172-31-3-136 ~]$ curl http://169.254.169.254/latest/meta-data/local-ipv4
172.31.3.136[ec2-user@ip-172-31-3-136 ~]$
```



Lokale IPV vier und wir erhalten die lokale IPV vier

↓↓↓↓↓

107. AWS EC2 Instance Metadata

```
ip-172-31-3-136.eu-west-3.compute.internal[ec2-user@ip-172-31-3-136 ~]$ curl http://169.254.169.254/latest/meta-data/iam/
info
security-credentials/[ec2-user@ip-172-31-3-136 ~]$ curl http://169.254.169.254/latest/meta-data/iam/security-credentials
[ec2-user@ip-172-31-3-136 ~]$ curl http://169.254.169.254/latest/meta-data/iam/security-credentials/
MyFirstEC2Role[ec2-user@ip-172-31-3-136 ~]$ curl http://169.254.169.254/latest/meta-data/iam/security-credentials/MyFirstEC2Role
{
    "Code" : "Success",
    "LastUpdated" : "2018-09-24T10:11:17Z",
    "Type" : "AWS-HMAC",
    "AccessKeyId" : "ASIAVUJTFK3Q2PW4H5JF",
    "SecretAccessKey" : "KqlccsxSseYKiGu+nNC1hM65/T90LSK7ssyec4/H",
    "Token" : "FQoGZXIVYDzEcwadC7swnGeeWyl8lg6hSkxA1qGPXJHK7eSxtLALMtC6IillNi6eM+WZ5FqC4bs+aEm/xjUm8SF9fi70XJQ0k10kMmvQSPaMV0s6o16I2cc23
oxw7TGFuPH///5HyIAkYW0ybNGHxbk0Q7f0oA01x8SSNULhNz0Pv+ScfbaMfGLueM9UypF7nKIgHqDMoJp4kSkEd40TfLq0m68QsXD02tYoE5gX1HoroTT8wDvSg9VZ0XUK19
//Y8v10kjDKPLjx9v0SLg/[PNQDU6fzBsAcshyPlH5/KYDphI62npFf00cab/iVGwcdCjdY/xhwAn5r7cGDJ8Ro9+uwDYj0k3ymnfGBvp+Zw29T0kGEypo7mvuy5Kaox4DA
<mdmN8ud7j8rZ6IQuP3UnaC1SR4b0IfcMnCcYkB2lJQ0CjhFkxau/gowzAwSUrsridIGl4bGj9D2uVvkYE7LwoRgn6r93psEXidzfRrMicwlumD+GNY7CwgvIpWpTQlqliDE
-1fpQ07Hmv9bEJL6boWmBqu9ID5/LYEtdy4INvWfPmxlvIyrrq9DwQPRt7PvlZ2KhEKFNrSWE98uErEpcDK0o2vCi3QU=",
    "Expiration" : "2018-09-24T16:15:54Z"
}[ec2-user@ip-172-31-3-136 ~]$ curl http://169.254.169.254/latest/meta-data/
```



URL hier, 169. 254. , und so weiter.



```
~/.aws  cat credentials
[default]
aws_access_key_id = ytU9W77n2jatdxMp
aws_secret_access_key = zspKxgTxjncmxkQ3sPraY6BfH88yNq552KBExMF
[my-other-aws-account]
aws_access_key_id = Dkn86Vv9q7JE8hHn
aws_secret_access_key = hgsJgwVR7WhRyu7z84rRAFDkPqd6hMUX2TcnPUA
~/.aws  cat config
[default]
region = eu-west-3
[profile my-other-aws-account]
region = us-west-2
~/.aws  aws s3 ls --profile my-other-aws-account
```



Das ist es also, es ist nicht etwas, das Sie für die Prüfung wissen sollten, aber es ist etwas,

Überprüfen

MFA with CLI

- To use MFA with the CLI, you must create a temporary session
 - To do so, you must run the STS GetSessionToken API call
-
- `aws sts get-session-token --serial-number arn-of-the-mfa-device --token-code code-from-token --duration-seconds 3600`

```
{  
  "Credentials": {  
    "SecretAccessKey": "secret-access-key",  
    "SessionToken": "temporary-session-token",  
    "Expiration": "expiration-date-time",  
    "AccessKeyId": "access-key-id"  
  }  
}
```

Schauen wir uns also



© Stephane Maarek

Überprüfen

```
x ~ aws sts get-session-token --serial-number arn:aws:iam::387124123361:mfa/stephane --token-code 828463sts
```



wir zum Generieren dieser temporären Sitzungstoken verwenden,
STS GetSession

Überprüfen

AWS SDK Overview

- We have to use the AWS SDK when coding against AWS Services such as DynamoDB
- Fun fact... the AWS CLI uses the Python SDK (boto3)
- The exam expects you to know when you should use an SDK
- We'll practice the AWS SDK when we get to the Lambda functions
- Good to know: if you don't specify or configure a default region, then us-east-1 will be chosen by default

API-Aufrufe auszugeben, und das kann

Überprüfen

© Stephane Maarek



AWS Limits (Quotas)

- API Rate Limits

- `DescribeInstances` API for EC2 has a limit of 100 calls per seconds
- `GetObject` on S3 has a limit of 5500 GET per second per prefix
- For Intermittent Errors: implement Exponential Backoff
- For Consistent Errors: request an API throttling limit increase

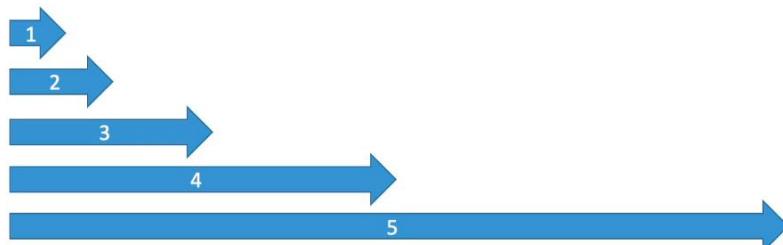
- Service Quotas (Service Limits)

- Running On-Demand Standard Instances: 1152 vCPU
- You can request a service limit increase by opening a ticket
- You can request a service quota increase by using the Service Quotas API



Exponential Backoff (any AWS service)

- If you get `ThrottlingException` intermittently, use exponential backoff
- Retry mechanism included in SDK API calls
- Must implement yourself if using the API as is or in specific cases



AWS CLI Credentials Provider Chain

- The CLI will look for credentials in this order
1. Command line options – --region, --output, and --profile
 2. Environment variables – AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY, and AWS_SESSION_TOKEN
 3. CLI credentials file –aws configure
~/.aws/credentials on Linux / Mac & C:\Users\user\.aws\credentials on Windows
 4. CLI configuration file – aws configure
~/.aws/config on Linux / macOS & C:\Users\USERNAME\.aws\config on Windows
 5. Container credentials – for ECS tasks
 6. Instance profile credentials – for EC2 Instance Profiles



© Stephane Maarek

sein, die ich Ihnen sehr bald erklären werde.

Über Sammy

AWS SDK Default Credentials Provider Chain

- The Java SDK (example) will look for credentials in this order
1. Environment variables –
AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY
 2. Java system properties – aws.accessKeyId and aws.secretKey
 3. The default credential profiles file – ex at: ~/.aws/credentials, shared by many SDK
 4. Amazon ECS container credentials – for ECS containers
 5. Instance profile credentials– used on EC2 instances



© Stephane Maarek

Also, was

Über Sammy



AWS Credentials Scenario

- An application deployed on an EC2 instance is using environment variables with credentials from an IAM user to call the Amazon S3 API.
- The IAM user has S3FullAccess permissions.
- The application only uses one S3 bucket, so according to best practices:
 - An IAM Role & EC2 Instance Profile was created for the EC2 instance
 - The Role was assigned the minimum permissions to access that one S3 bucket
- The IAM Instance Profile was assigned to the EC2 instance, but it still had access to all S3 buckets. Why?
the credentials chain is still giving priorities to the environment variables

© Stephane Maarek

die Sie zuvor festgelegt haben, weiterhin Prioritäten zuweist.

by [Stephane Maarek](#)

AWS Credentials Best Practices

- Overall, NEVER EVER STORE AWS CREDENTIALS IN YOUR CODE
- Best practice is for credentials to be inherited from the credentials chain
- If using working within AWS, use IAM Roles
 - => EC2 Instances Roles for EC2 Instances
 - => ECS Roles for ECS tasks
 - => Lambda Roles for Lambda functions
- If working outside of AWS, use environment variables / named profiles

© Stephane Maarek

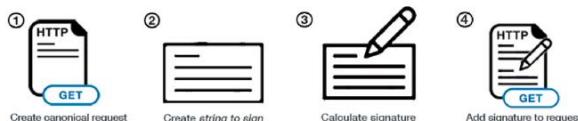
Hoffe das war hilfreich.

by [Stephane Maarek](#)



Signing AWS API requests

- When you call the AWS HTTP API, you sign the request so that AWS can identify you, using your AWS credentials (access key & secret key)
- Note: some requests to Amazon S3 don't need to be signed
- If you use the SDK or CLI, the HTTP requests are signed for you
- You should sign an AWS HTTP request using Signature v4 (SigV4)



© Stephane Maarek

auf der Website erfahren Sie, wie Sie dies unterschreiben.

Über Sammy

SigV4 Request examples

- HTTP Header option

```
GET https://iam.amazonaws.com?Action=ListUsers&Version=2010-05-08 HTTP/1.1
Authorization: AWS4-HMAC-SHA256 Credential=AKIDEXAMPLE/20150830/us-east-1/iam/aws4_request,
SignedHeaders=content-type;host;x-amz-date,
Signature=5d672d79c15b13162d9279b0855cfba6789a8edb4c82c400e06b5924a6f2b5d7
content-type: application/x-www-form-urlencoded; charset=utf-8
host: iam.amazonaws.com
x-amz-date: 20150830T123600Z
```



- Query String option (ex: S3 pre-signed URLs)

```
GET https://iam.amazonaws.com?Action=ListUsers&Version=2010-05-08&
X-Amz-Algorithm=AWS4-HMAC-SHA256&
X-Amz-Credential=AKIDEXAMPLE%2F20150830%2Fus-east-1%2Fiam%2Faws4_request&
X-Amz-Date=20150830T123600Z&X-Amz-Expires=60&X-Amz-SignedHeaders=content-type%3Bhost&
X-Amz-Signature=37ac2f4fdde00b0ac9bd9eadeb459b1bbe224158d66e7ae5fcadb70b2d181d02 HTTP/1.1
content-type: application/x-www-form-urlencoded; charset=utf-8
host: iam.amazonaws.com
```

© Stephane Maarek

Jetzt zeige ich Ihnen diese URL nur

Über Sammy

S3 MFA-Delete

- MFA (multi factor authentication) forces user to generate a code on a device (usually a mobile phone or hardware) before doing important operations on S3
- To use MFA-Delete, enable Versioning on the S3 bucket
- You will need MFA to
 - permanently delete an object version
 - suspend versioning on the bucket
- You won't need MFA for
 - enabling versioning
 - listing deleted versions
- Only the bucket owner (root account) can enable/disable MFA-Delete
- MFA-Delete currently can only be enabled using the CLI

Dafür müssen Sie Root-Anmeldeinformationen verwenden, und es gibt

© Stephane Maarek

116. S3 Access Logs

S3 Access Logs

- For audit purpose, you may want to log all access to S3 buckets
- Any request made to S3, from any account, authorized or denied, will be logged into another S3 bucket
- That data can be analyzed using data analysis tools...
- Or Amazon Athena as we'll see later in this section!
- The log format is at:
<https://docs.aws.amazon.com/AmazonS3/latest/dev/LogFormat.html>

Wenn Sie also interessiert sind, wie wir dieses Protokoll lesen

My-bucket

Logging Bucket

Log all requests

requests

←

Stephane Maarek 0:44 / 1:49

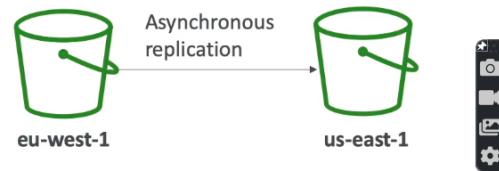
S3 Access Logs: Warning

- Do not set your logging bucket to be the monitored bucket
- It will create a logging loop, and your bucket will grow in size exponentially



S3 Replication (CRR & SRR)

- Must enable versioning in source and destination
 - Cross Region Replication (CRR)
 - Same Region Replication (SRR)
 - Buckets can be in different accounts
 - Copying is asynchronous
 - Must give proper IAM permissions to S3
- CRR - Use cases: compliance, lower latency access, replication across accounts
- SRR – Use cases: log aggregation, live replication between production and test accounts



S3 Replication – Notes

- After activating, only new objects are replicated (not retroactive)
- For DELETE operations:
 - If you delete without a version ID, it adds a delete marker; not replicated
 - If you delete with a version ID, it deletes in the source, not replicated
- There is no “chaining” of replication
 - If bucket 1 has replication into bucket 2, which has replication into bucket 3
 - Then objects created in bucket 1 are not replicated to bucket 3



© Stephane Maarek

sodass wir Ihre Replikation nicht verketten können.

120. S3 Pre-signed URLs

S3 pre-signed URLs

• Can generate pre-signed URLs using SDK or CLI

- For downloads (easy, can use the CLI)
- For uploads (harder; must use the SDK)

• Valid for a default of 3600 seconds, can change timeout with --expires-in [TIME_BY_SECONDS] argument

• Users given a pre-signed URL inherit the permissions of the person who generated the URL for GET / PUT

• Examples :

- Allow only logged-in users to download a premium video on your S3 bucket
- Allow an ever changing list of users to download files by generating URLs dynamically
- Allow temporarily a user to upload a file to a precise location in our bucket

Datei an einen bestimmten Ort in unseren Buckets hochzuladen.

© Stephane Maarek 1x 1:22 / 1:36

CC G

The screenshot shows a code editor interface with the following details:

- EXPLORER** sidebar: Shows a tree view of AWS services: CODE (api-gateway, cli, cloudformation, ebs, ec2-fundamentals, efs, kms, route53, s3), s3-advanced (athena-s3-access-logs.sql, Icon, mfa-delete.sh, pre-signed-url.sh), sqs, ssm, Icon, OUTLINE, and NPM SCRIPTS.
- PRE-SIGNED-URL SH** tab: Active tab showing a shell script for generating pre-signed URLs.
- CODE** tab: Shows the script content:

```
1
2
3 # do not forget to region parameter! (make sure it's the proper
4 aws s3 presign s3://mybucket/myobject --region my-region
5
6 # add a custom expiration time
7 aws s3 presign s3://mybucket/myobject --expires-in 300 --region
8 my-region
9
10 # IF YOU ARE GETTING ISSUES
11
12 # set the proper signature version in order not to get issues
when generating URLs for encrypted files
13 aws configure set default.s3.signature_version s3v4
14
```

A tooltip message is displayed in the center of the editor area: "Sie Probleme bekommen und nicht nur diesen Befehl verwenden, können Sie loslegen."

Bottom status bar: Ln 12, Col 103 (52 selected) Spaces: 4 UTF-8 LF Shell Script

S3 Storage Classes

- Amazon S3 Standard - General Purpose
 - Amazon S3 Standard-Infrequent Access (IA)
 - Amazon S3 One Zone-Infrequent Access
 - Amazon S3 Intelligent Tiering
 - Amazon Glacier
 - Amazon Glacier Deep Archive
-
- Amazon S3 Reduced Redundancy Storage (deprecated - omitted)



S3 Standard – General Purpose

- High durability (99.99999999%) of objects across multiple AZ
 - If you store 10,000,000 objects with Amazon S3, you can on average expect to incur a loss of a single object once every 10,000 years
 - 99.9% Availability over a given year
 - Sustain 2 concurrent facility failures
-
- Use Cases: Big Data analytics, mobile & gaming applications, content distribution...

© Stephane Maarek

Dies ist im Grunde alles, was wir bisher verwendet haben.

by [Stephane Maarek](#)

S3 Standard – Infrequent Access (IA)

- Suitable for data that is less frequently accessed, but requires rapid access when needed
 - High durability (99.99999999%) of objects across multiple AZs
 - 99.9% Availability
 - Low cost compared to Amazon S3 Standard
 - Sustain 2 concurrent facility failures
-
- Use Cases: As a data store for disaster recovery, backups...

© Stephane Maarek

alle Dateien sein, auf die Sie voraussichtlich weniger häufig zugreifen werden.

by [Stephane Maarek](#)

S3 One Zone - Infrequent Access (IA)

- Same as IA but data is stored in a single AZ
 - High durability (99.99999999%) of objects in a single AZ; data lost when AZ is destroyed
 - 99.5% Availability
 - Low latency and high throughput performance
 - Supports SSL for data at transit and encryption at rest
 - Low cost compared to IA (by 20%)
-
- Use Cases: Storing secondary backup copies of on-premise data, or storing data you can recreate

um das Bild in S3 General Purpose und



S3 Intelligent Tiering

- Same low latency and high throughput performance of S3 Standard
- Small monthly monitoring and auto-tiering fee
- Automatically moves objects between two access tiers based on changing access patterns
- Designed for durability of 99.99999999% of objects across multiple Availability Zones
- Resilient against events that impact an entire Availability Zone
- Designed for 99.9% availability over a given year

Amazon Glacier

- Low cost object storage meant for archiving / backup
- Data is retained for the longer term (10s of years)
- Alternative to on-premise magnetic tape storage
- Average annual durability is 99.999999999%
- Cost per storage per month (\$0.004 / GB) + retrieval cost
- Each item in Glacier is called "Archive" (up to 40TB)
- Archives are stored in "Vaults"



Und Archive werden gespeichert, nicht in

Amazon Glacier & Glacier Deep Archive

- Amazon Glacier – 3 retrieval options:
 - Expedited (1 to 5 minutes)
 - Standard (3 to 5 hours)
 - Bulk (5 to 12 hours)
 - Minimum storage duration of 90 days
- Amazon Glacier Deep Archive – for long term storage – cheaper:
 - Standard (12 hours)
 - Bulk (48 hours)
 - Minimum storage duration of 180 days

Die minimale



S3 Storage Classes Comparison

	S3 Standard	S3 Intelligent-Tiering	S3 Standard-IA	S3 One Zone-IA	S3 Glacier	S3 Glacier Deep Archive
Designed for durability	99.999999999% (11 9's)					
Designed for availability	99.99%	99.9%	99.9%	99.5%	99.99%	99.99%
Availability SLA	99.9%	99%	99%	99%	99.9%	99.9%
Availability Zones	≥3	≥3	≥3	1	≥3	≥3
Minimum capacity charge per object	N/A	N/A	128KB	128KB	40KB	40KB
Minimum storage duration charge	N/A	30 days	30 days	30 days	90 days	180 days
Retrieval fee	N/A	N/A	per GB retrieved	per GB retrieved	per GB retrieved	per GB retrieved



© Stephane Maarek

auf der Anzahl der <https://aws.amazon.com/s3/storage-classes/>

122. S3 Storage Classes + Glacier

S3 Storage Classes – Price Comparison Example us-east-2

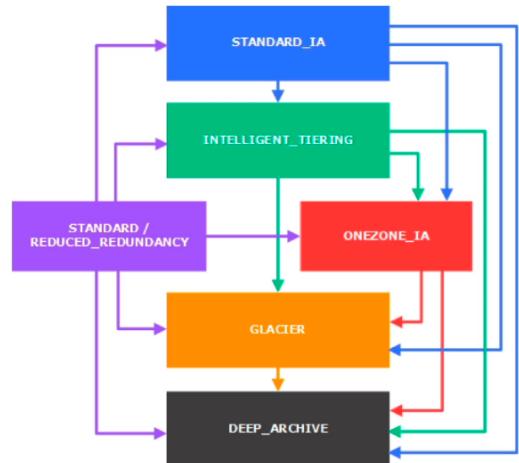
	S3 Standard	S3 Intelligent-Tiering	S3 Standard-IA	S3 One Zone-IA	S3 Glacier	S3 Glacier Deep Archive
Storage Cost (per GB per month)	\$0.023	\$0.0125 - \$0.023	\$0.0125	\$0.01	\$0.004 Minimum 90 days	\$0.00099 Minimum 180 days
Retrieval Cost (per 1000 requests)	GET \$0.0004	GET \$0.0004	GET \$0.001	GET \$0.001	GET \$0.0004 + Expedited - \$10.00 Standard - \$0.05 Bulk - \$0.025	GET \$0.0004 + Standard - \$0.10 Bulk - \$0.025
Time to retrieve	instantaneous	Instantaneous	Instantaneous	Instantaneous	Expedited (1 to 5 minutes) Standard (3 to 5 hours) Bulk (5 to 12 hours)	Standard (12 hours) Bulk (48 hours)
Monitoring Cost (per 1000 objects)		\$0.0025				

Objekten, da diese bei Bedarf

Stephane Maarek 1x 9:30 / 9:45

S3 – Moving between storage classes

- You can transition objects between storage classes
- For infrequently accessed object, move them to STANDARD_IA
- For archive objects you don't need in real-time, GLACIER or DEEP_ARCHIVE
- Moving objects can be automated using a lifecycle configuration



© Stephane Maarek

wissen, dass Sie in die Prüfung gehen.



124. S3 Lifecycle Rules

S3 Lifecycle Rules

- Transition actions: It defines when objects are transitioned to another storage class.
 - Move objects to Standard IA class 60 days after creation
 - Move to Glacier for archiving after 6 months
- Expiration actions: configure objects to expire (delete) after some time
 - Access log files can be set to delete after a 365 days
 - Can be used to delete old versions of files (if versioning is enabled)
 - Can be used to delete incomplete multi-part uploads
- Rules can be created for a certain prefix (ex - s3://mybucket/mp3/*)
- Rules can be created for certain objects tags (ex - Department: Finance)

Objekt-Tags erstellen lassen.

© Stephane Maarek 1x 2:14 / 5:17





S3 Lifecycle Rules – Scenario 1

- Your application on EC2 creates images thumbnails after profile photos are uploaded to Amazon S3. These thumbnails can be easily recreated, and only need to be kept for 45 days. The source images should be able to be immediately retrieved for these 45 days, and afterwards, the user can wait up to 6 hours. How would you design this?
- S3 source images can be on STANDARD, with a lifecycle configuration to transition them to GLACIER after 45 days.
- S3 thumbnails can be on ONEZONE_IA, with a lifecycle configuration to expire them (delete them) after 45 days.

© Stephane Maarek

um sie abzulaufen oder nach 45 Tagen zu löschen.

by Sunny

S3 Lifecycle Rules – Scenario 2

- A rule in your company states that you should be able to recover your deleted S3 objects immediately for 15 days, although this may happen rarely. After this time, and for up to 365 days, deleted objects should be recoverable within 48 hours.
- You need to enable S3 versioning in order to have object versions, so that “deleted objects” are in fact hidden by a “delete marker” and can be recovered
- You can transition these “noncurrent versions” of the object to S3_IA
- You can transition afterwards these “noncurrent versions” to DEEP_ARCHIVE

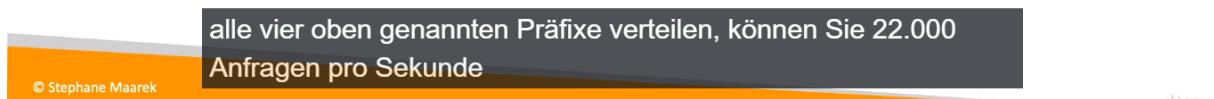
© Stephane Maarek

für 100 und 365 Tage.

by Sunny

S3 – Baseline Performance

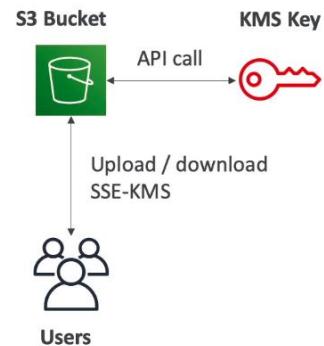
- Amazon S3 automatically scales to high request rates, latency 100-200 ms
- Your application can achieve at least 3,500 PUT/COPY/POST/DELETE and 5,500 GET/HEAD requests per second per prefix in a bucket.
- There are no limits to the number of prefixes in a bucket.
- Example (object path => prefix):
 - bucket/folder1/sub1/file => /folder1/sub1/
 - bucket/folder1/sub2/file => /folder1/sub2/
 - bucket/1/file => /1/
 - bucket/2/file => /2/
- If you spread reads across all four prefixes evenly, you can achieve 22,000 requests per second for GET and HEAD



126. S3 Performance

S3 – KMS Limitation

- If you use SSE-KMS, you may be impacted by the KMS limits
- When you upload, it calls the GenerateDataKey KMS API
- When you download, it calls the Decrypt KMS API
- Count towards the KMS quota per second (5500, 10000, 30000 req/s based on region)
- As of today, you cannot request a quota increase for KMS



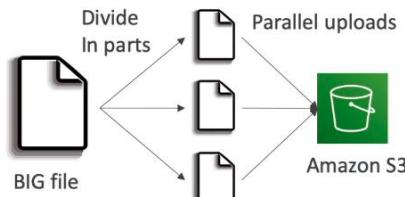
Dies bedeutet also, dass Sie gedrosselt werden, wenn Sie mehr



126. S3 Performance

S3 Performance

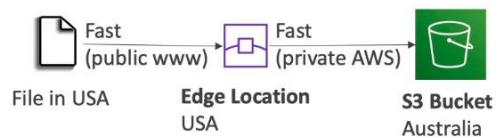
- Multi-Part upload:
 - recommended for files > 100MB, must use for files > 5GB
 - Can help parallelize uploads (speed up transfers)



schnelle private AWS-Netzwerk.

- S3 Transfer Acceleration (upload only)

- Increase transfer speed by transferring file to an AWS edge location which will forward the data to the S3 bucket in the target region
- Compatible with multi-part upload



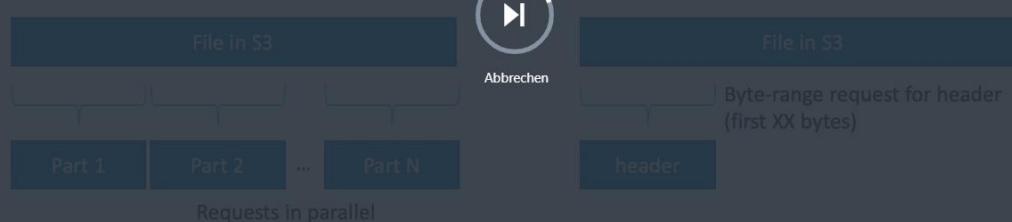
126. S3 Performance

S3 Performance – S3 Byte-Range Fetches

- Parallelize GETs by requesting specific byte ranges
- Better resilience in case of failures

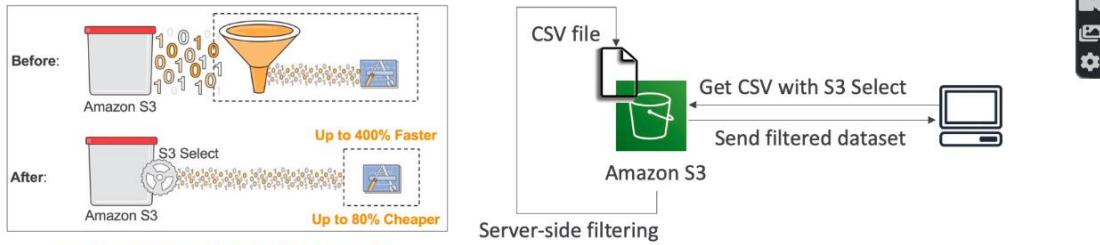
Can be used to speed up downloads

Jetzt kommt
127. S3 & Glacier Select
Can be used to retrieve only partial data (for example the head of a file)



S3 Select & Glacier Select

- Retrieve less data using SQL by performing **server side filtering**
- Can filter by rows & columns (simple SQL statements)
- Less network transfer, less CPU cost client-side



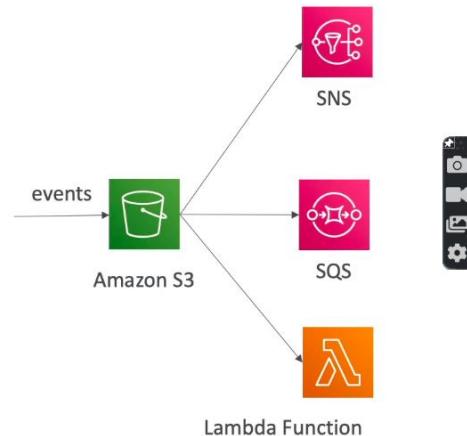
© Stephane Maarek

schneller. Das ist also großartig.

Überall

S3 Event Notifications

- S3:ObjectCreated, S3:ObjectRemoved, S3:ObjectRestore, S3:Replication...
 - Object name filtering possible (*.jpg)
 - Use case: generate thumbnails of images uploaded to S3
 - Can create as many "S3 events" as desired
-
- S3 event notifications typically deliver events in seconds but can sometimes take a minute or longer
 - If two writes are made to a single non-versioned object at the same time, it is possible that only a single event notification will be sent
 - If you want to ensure that an event notification is sent for every successful write, you can enable versioning on your bucket.



© Stephane Maarek

Schauen wir uns also in der Konsole an, wie

Überall

AWS Athena



- Serverless service to perform analytics directly against S3 files
- Uses SQL language to query the files
- Has a JDBC / ODBC driver
- Charged per query and amount of data scanned
- Supports CSV, JSON, ORC, Avro, and Parquet (built on Presto)
- Use cases: Business intelligence / analytics / reporting, analyze & query VPC Flow Logs, ELB Logs, CloudTrail trails, etc...
- Exam Tip: Analyze data directly on S3 => use Athena

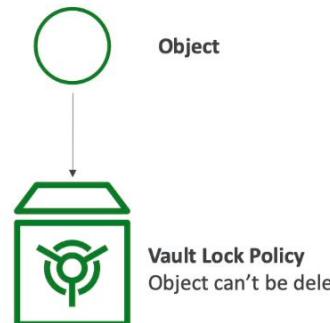
© Stephane Maarek

"Hey, wie können wir Daten direkt in S3 analysieren?

15:00 min

S3 Object Lock & Glacier Vault Lock

- S3 Object Lock
 - Adopt a WORM (Write Once Read Many) model
 - Block an object version deletion for a specified amount of time
- Glacier Vault Lock
 - Adopt a WORM (Write Once Read Many) model
 - Lock the policy for future edits (can no longer be changed)
 - Helpful for compliance and data retention



© Stephane Maarek

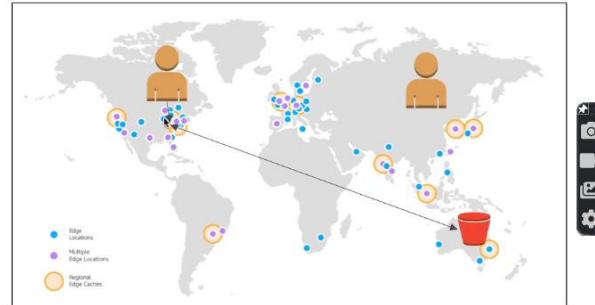
gelöscht werden, und Sie haben die Gewissheit, dass es von niemandem

15:00 min

AWS CloudFront



- Content Delivery Network (CDN)
- Improves read performance, content is cached at the edge
- 216 Point of Presence globally (edge locations)
- DDoS protection, integration with Shield, AWS Web Application Firewall
- Can expose external HTTPS and can talk to internal HTTPS backends



Source: <https://aws.amazon.com/cloudfront/features/?nc=sn&loc=2>

© Stephane Maarek

Ein anderer Benutzer, möglicherweise in Asien, spricht also

133. CloudFront - Overview

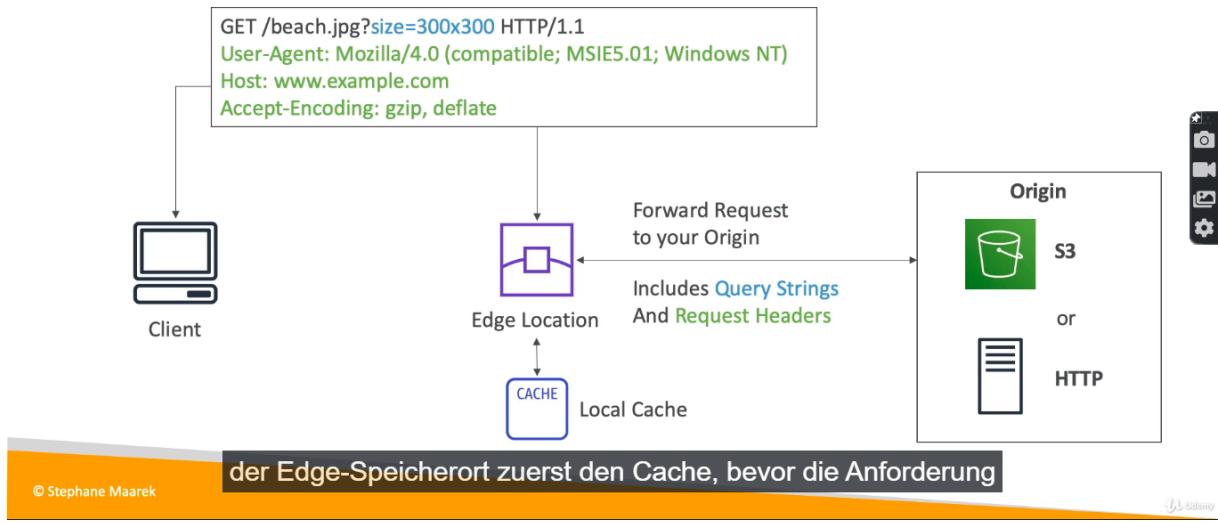
CloudFront – Origins

- S3 bucket
 - For distributing files and caching them at the edge
 - Enhanced security with CloudFront Origin Access Identity (OAI)
 - CloudFront can be used as an ingress (to upload files to S3)
- Custom Origin (HTTP)
 - Application Load Balancer
 - EC2 instance
 - S3 website (must first enable the bucket as a static S3 website)
 - Any HTTP backend you want

Und wir könnten jedes gewünschte HTTP-Backend sein,

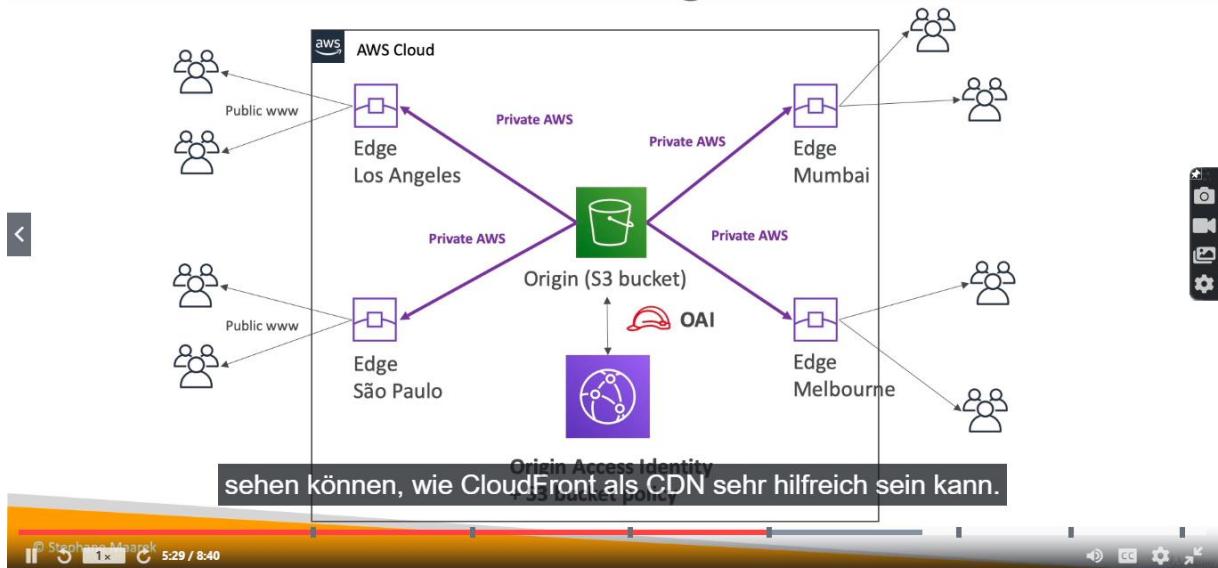
Stephane Maarek 3:17 / 8:40

CloudFront at a high level

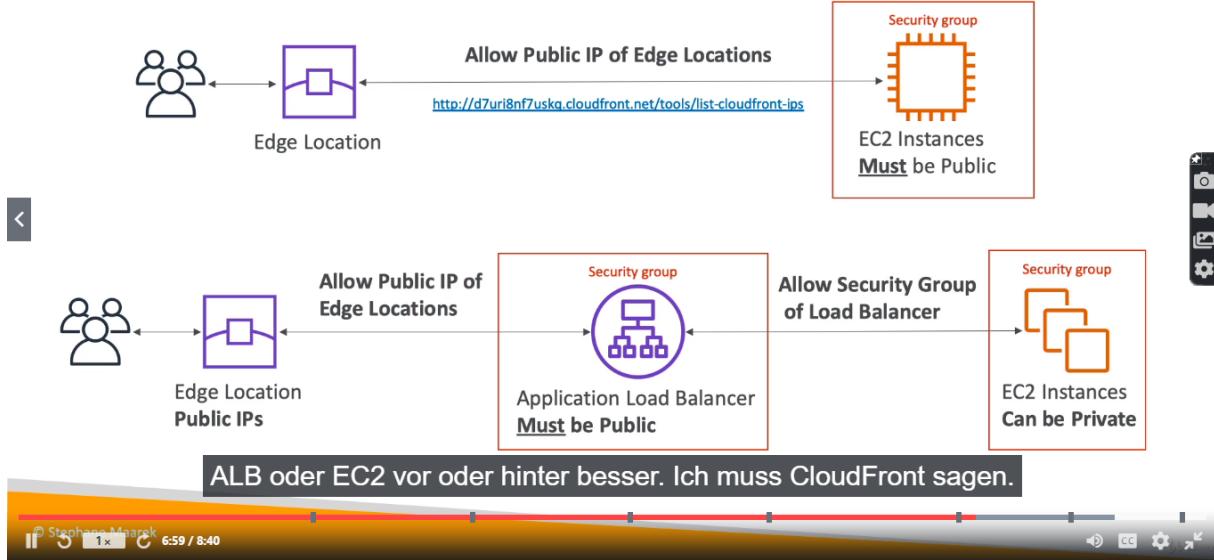


133. CloudFront - Overview

CloudFront – S3 as an Origin



CloudFront – ALB or EC2 as an origin



CloudFront Geo Restriction

- You can restrict who can access your distribution
 - Whitelist: Allow your users to access your content only if they're in one of the countries on a list of approved countries.
 - Blacklist: Prevent your users from accessing your content if they're in one of the countries on a blacklist of banned countries.
- The “country” is determined using a 3rd party Geo-IP database
- Use case: Copyright Laws to control access to content

beispielsweise Frankreich einschränken, wenn Sie Inhalte in Amerika haben.

CloudFront vs S3 Cross Region Replication

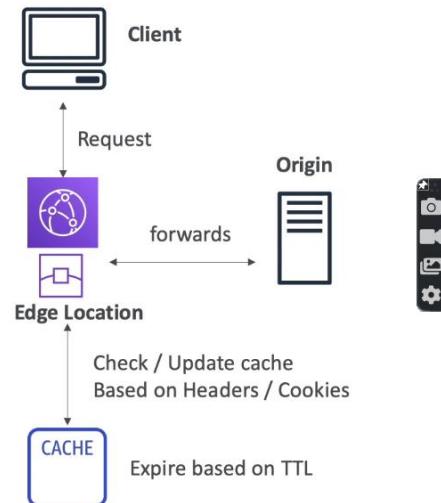
- CloudFront:
 - Global Edge network
 - Files are cached for a TTL (maybe a day)
 - Great for static content that must be available everywhere
- S3 Cross Region Replication:
 - Must be setup for each region you want replication to happen
 - Files are updated in near real-time
 - Read only
 - Great for dynamic content that needs to be available at low-latency in few regions

Regionen mit geringer Latenz verfügbar sein muss.

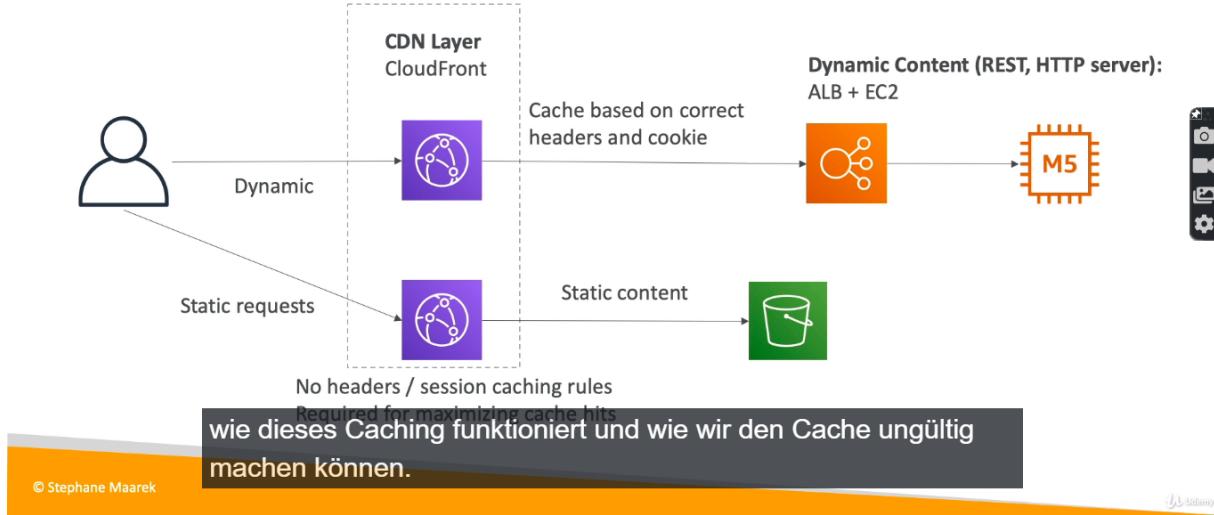


CloudFront Caching

- Cache based on
 - Headers
 - Session Cookies
 - Query String Parameters
- The cache lives at each CloudFront Edge Location
- You want to maximize the cache hit rate to minimize requests on the origin
- Control the TTL (0 seconds to 1 year), can be set by the origin using the Cache-Control header, Expires header...
- You can invalidate part of the cache using the CreateInvalidation API



CloudFront – Maximize cache hits by separating static and dynamic distributions



CloudFront Geo Restriction

- You can restrict who can access your distribution
 - Whitelist: Allow your users to access your content only if they're in one of the countries on a list of approved countries.
 - Blacklist: Prevent your users from accessing your content if they're in one of the countries on a blacklist of banned countries.
- The “country” is determined using a 3rd party Geo-IP database
- Use case: Copyright Laws to control access to content

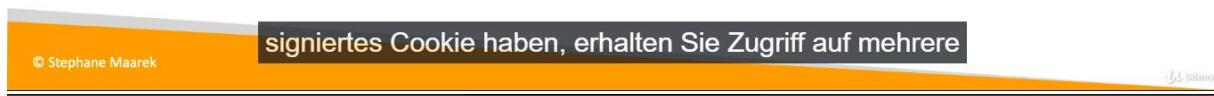
geografische Einschränkung wären Urheberrechtsgesetze zur Kontrolle des Zugriffs auf Inhalte.

© Stephane Maarek

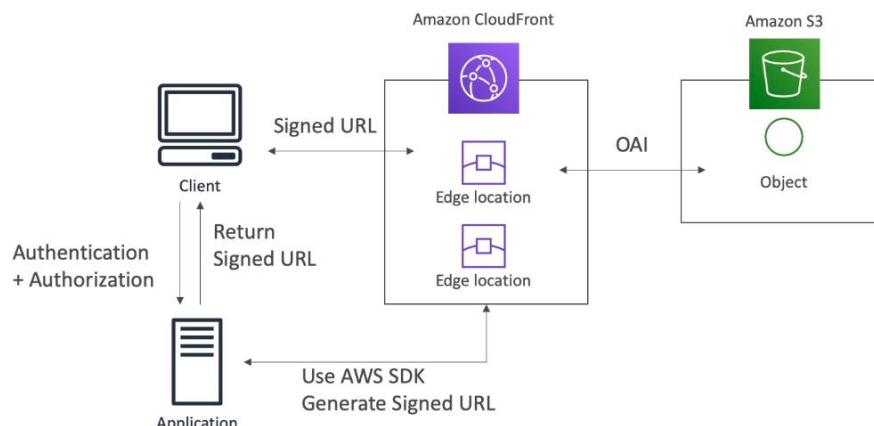
by Stephane Maarek

CloudFront Signed URL / Signed Cookies

- You want to distribute paid shared content to premium users over the world
- We can use CloudFront Signed URL / Cookie. We attach a policy with:
 - Includes URL expiration
 - Includes IP ranges to access the data from
 - Trusted signers (which AWS accounts can create signed URLs)
- How long should the URL be valid for?
 - Shared content (movie, music): make it short (a few minutes)
 - Private content (private to the user): you can make it last for years
- Signed URL = access to individual files (one signed URL per file)
- Signed Cookies = access to multiple files (one signed cookie for many files)

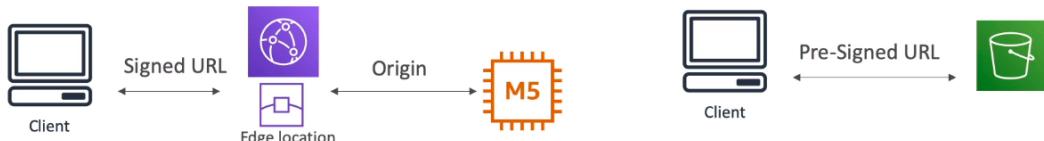


CloudFront Signed URL Diagram



CloudFront Signed URL vs S3 Pre-Signed URL

- CloudFront Signed URL:
 - Allow access to a path, no matter the origin
 - Account wide key-pair, only the root can manage it
 - Can filter by IP, path, date, expiration
 - Can leverage caching features
- S3 Pre-Signed URL:
 - Issue a request as the person who pre-signed the URL
 - Uses the IAM key of the signing IAM principal
 - Limited lifetime



© Stephane Maarek

müssen Sie eine signierte URL verwenden, da

↳ [zum Beitrag](#)

What is Docker?



- Docker is a software development platform to deploy apps
- Apps are packaged in containers that can be run on any OS
- Apps run the same, regardless of where they're run
 - Any machine
 - No compatibility issues
 - Predictable behavior
 - Less work
 - Easier to maintain and deploy
 - Works with any language, any OS, any technology

AWS Certified Developer © Stephane Maarek

ich wirklich sehr, sehr toll finde.

↳ [zum Beitrag](#)

139. What is Docker?

Where Docker images are stored?

- Docker images are stored in Docker Repositories

- Public: Docker Hub <https://hub.docker.com/>

- Find base images for many technologies or OS:
 - Ubuntu
 - MySQL
 - NodeJS, Java...

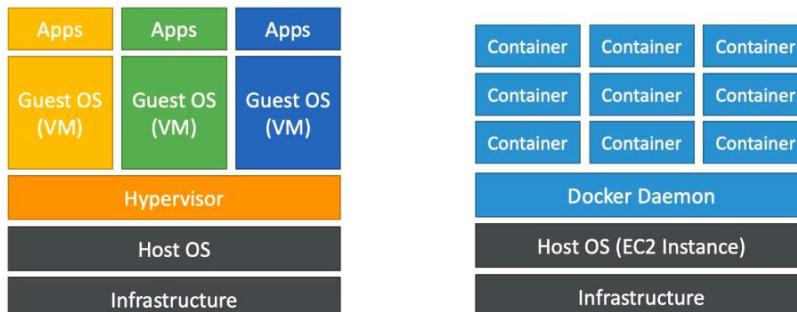
- Private: Amazon ECR (Elastic Container Registry)

in dem Sie Ihre persönlichen Bilder speichern.

139. What is Docker?

Docker versus Virtual Machines

- Docker is "sort of" a virtualization technology, but not exactly
- Resources are shared with the host => many containers on one server



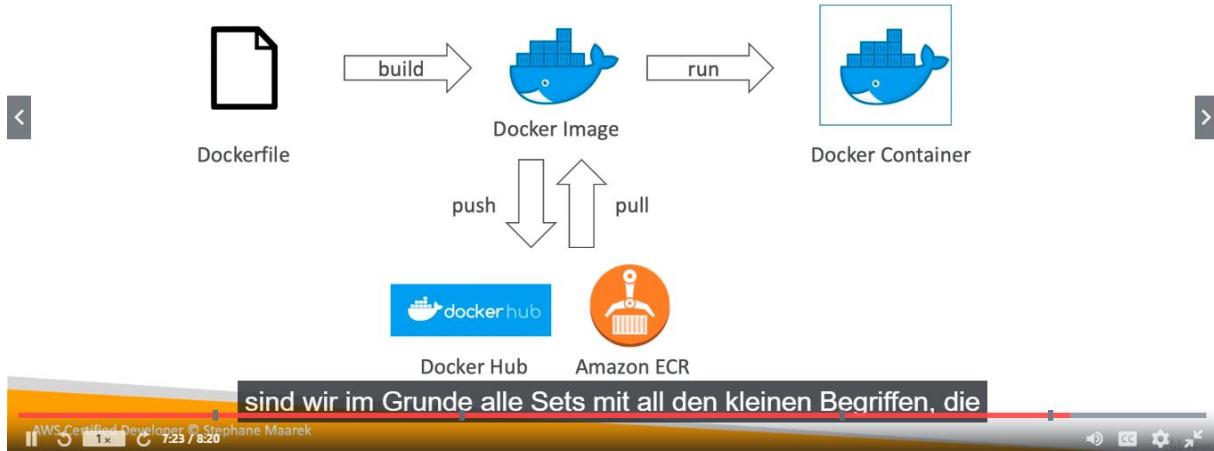
veranschaulichen, die auf einer höheren Ebene

The screenshot shows the Docker Hub 'Getting Started with Docker' page. At the top, there are tabs for 'Developer', 'IT', and 'Business Leaders'. Below the tabs, the title 'Docker for Developers' is displayed. A text block explains that building and deploying new applications is faster with containers, as Docker containers wrap up software and its dependencies into a standardized unit for software development. It also mentions that Docker containers are backed by Docker tools and APIs, allowing for faster onboarding and the ability to run code locally. A sidebar on the right provides download links for Mac and Windows, and a 'Try a Free Lab' button. A callout box highlights the text 'musst Docker installieren, wenn du gehen'.

139. What is Docker?

Getting Started with Docker

- Download Docker at: <https://www.docker.com/get-started>



Docker Containers Management

- To manage containers, we need a container management platform

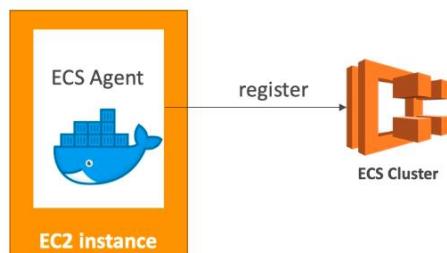
- Three choices:
- ECS: Amazon's own platform
- Fargate: Amazon's own Serverless platform
- EKS: Amazon's managed Kubernetes (open source)

bessere Vorstellung davon, wie Docker ist. Ich weiß,



ECS Clusters Overview

- ECS Clusters are logical grouping of EC2 instances
- EC2 instances run the ECS agent (Docker container)
- The ECS agents registers the instance to the ECS cluster
- The EC2 instances run a special AMI, made specifically for ECS

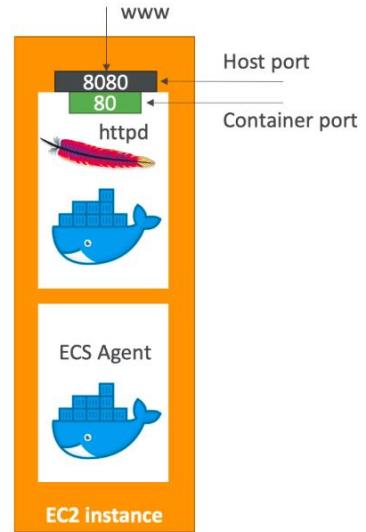


unsere Instanz sehr einfach im ECS-Cluster registrieren.



ECS Task Definitions

- Tasks definitions are metadata in JSON form to tell ECS how to run a Docker Container
- It contains crucial information around:
 - Image Name
 - Port Binding for Container and Host
 - Memory and CPU required
 - Environment variables
 - Networking information



Okay, genug Theorie, lass uns zur Praxis gehen.

AWS Certified Developer © Stephane Maarek

Überblick

ECS Service

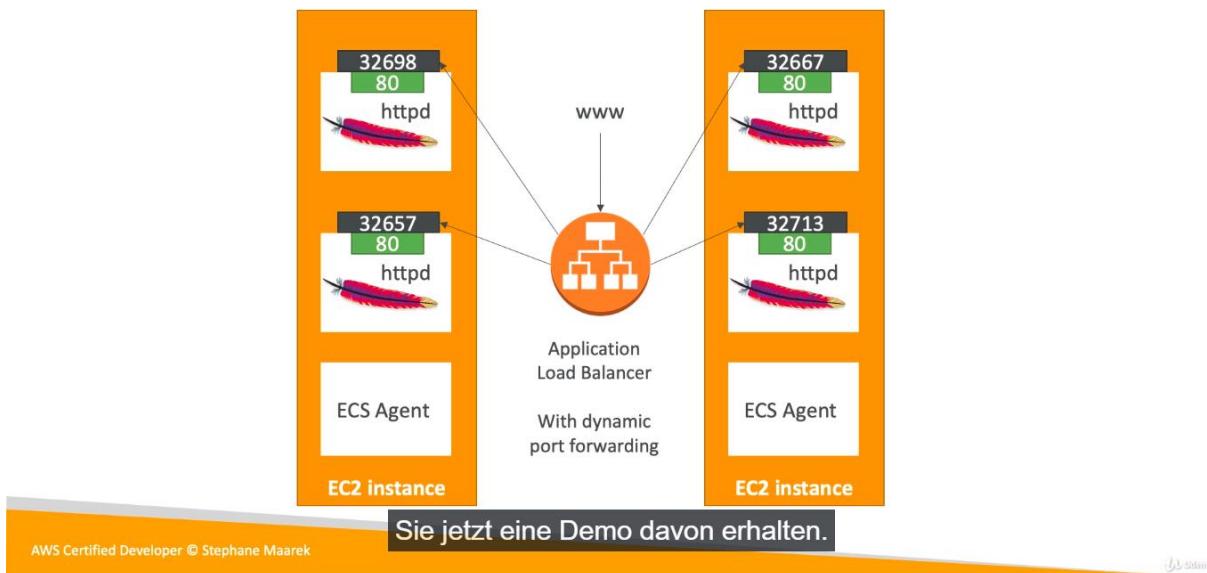
- ECS Services help define how many tasks should run and how they should be run
- They ensure that the number of tasks desired is running across our fleet of EC2 instances.
- They can be linked to ELB / NLB / ALB if needed
- Let's make our first service!

Machen wir also unseren ersten Service OK.

AWS Certified Developer © Stephane Maarek

Überblick

ECS Service with Load Balancer



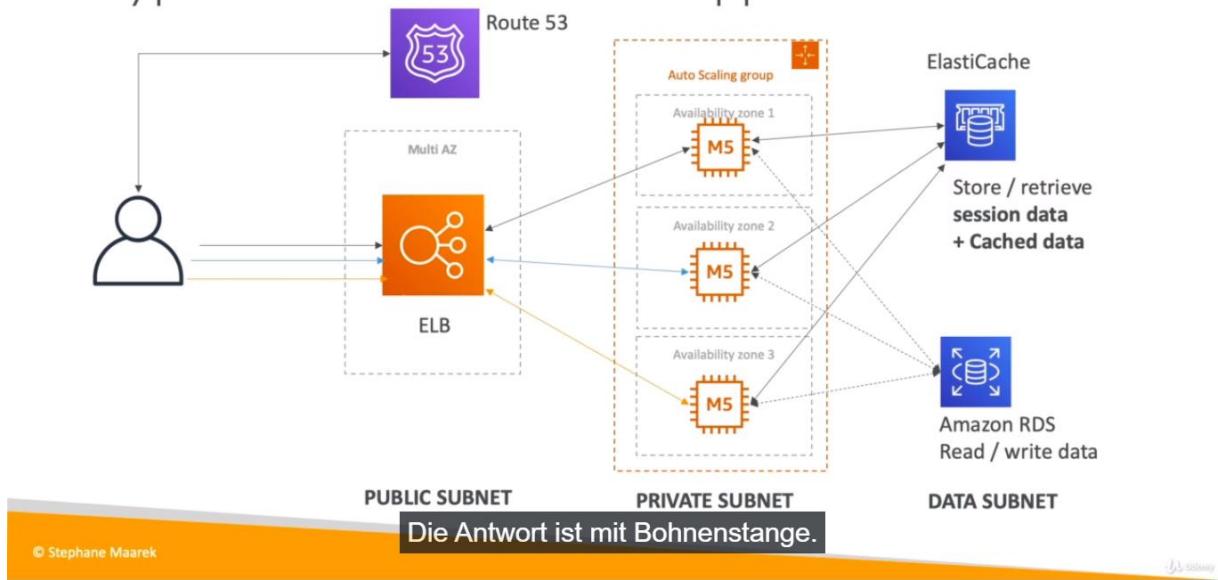
ECR



- So far we've been using Docker images from Docker Hub (public)
- ECR is a private Docker image repository
- Access is controlled through IAM (permission errors => policy)
- AWS CLI v1 login command (may be asked at the exam)
 - \$(aws ecr get-login --no-include-email --region eu-west-1)
- AWS CLI v2 login command (newer; may also be asked at the exam - pipe)
 - aws ecr get-login-password --region eu-west-1 | docker login --username AWS --password-stdin 1234567890.dkr.ecr.eu-west-1.amazonaws.com
- Docker Push & Pull:
 - docker push 1234567890.dkr.ecr.eu-west-1.amazonaws.com/demo:latest
 - docker pull 1234567890.dkr.ecr.eu-west-1.amazonaws.com/demo:latest

vollständige Bild-URL ein, einschließlich des ECR-Repositorys, und für Docker Pull.

Typical architecture: Web App 3-tier



Developer problems on AWS

- Managing infrastructure
 - Deploying Code
 - Configuring all the databases, load balancers, etc
 - Scaling concerns
-
- Most web apps have the same architecture (ALB + ASG)
 - All the developers want is for their code to run!
 - Possibly, consistently across different applications and environments

© Stephane Maarek

und Programmiersprachen tun.

CC BY-SA

AWS Elastic Beanstalk Overview



- Elastic Beanstalk is a developer centric view of deploying an application on AWS
- It uses all the component's we've seen before: EC2, ASG, ELB, RDS, etc...
- But it's all in one view that's easy to make sense of!
- We still have full control over the configuration
- Beanstalk is free but you pay for the underlying instances

© Stephane Maarek Wenn Sie beispielsweise eine EC2-Instanz b3: 30sec

Elastic Beanstalk

- Managed service
 - Instance configuration / OS is handled by Beanstalk
 - Deployment strategy is configurable but performed by Elastic Beanstalk
- Just the application code is the responsibility of the developer
- Three architecture models:
 - Single Instance deployment: good for dev
 - LB + ASG: great for production or pre-production web applications
 - ASG only: great for non-web apps in production (workers, etc..)

© Stephane Maarek ist ideal, wenn Sie einige Nachrichten einer Warteschlange verarbeiten möchten . b3: 30sec

Elastic Beanstalk

- Elastic Beanstalk has three components
 - Application
 - Application version: each deployment gets assigned a version
 - Environment name (dev, test, prod...): free naming
- You deploy application versions to environments and can promote application versions to the next environment
- Rollback feature to previous application version
- Full control over lifecycle of environments



Sinn machen, sobald wir in die Praxis gehen.

Elastic Beanstalk

- Support for many platforms:
 - Go
 - Java SE
 - Java with Tomcat
 - .NET on Windows Server with IIS
 - Node.js
 - PHP
 - Python
 - Ruby
 - Packer Builder
- Single Container Docker
- Multicontainer Docker
- Preconfigured Docker
- If not supported, you can write your custom platform (advanced)

und unsere erste Beanstalk-Anwendung zu erstellen.

Elastic Beanstalk Deployment Modes



156. Beanstalk Deployment Modes

Beanstalk Deployment Options for Updates

- All at once (**deploy all in one go**) – fastest, but instances aren't available to serve traffic for a bit (downtime)
- Rolling: update a few instances at a time (bucket), and then move onto the next bucket once the first bucket is healthy
- Rolling with additional batches: like rolling, but spins up new instances to move the batch (so that the old application is still available)
- Immutable: spins up new instances in a new ASG, deploys version to these instances, and then swaps all the instances when everything is healthy

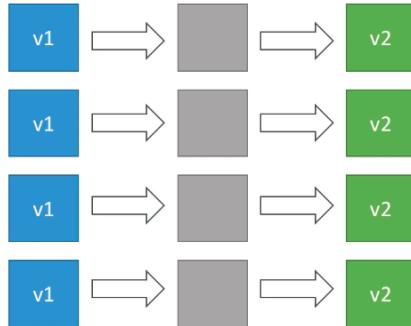
Das ist also ein bisschen hoch und Sie haben



Elastic Beanstalk Deployment

All at once

- Fastest deployment
- Application has downtime
- Great for quick iterations in development environment
- No additional cost

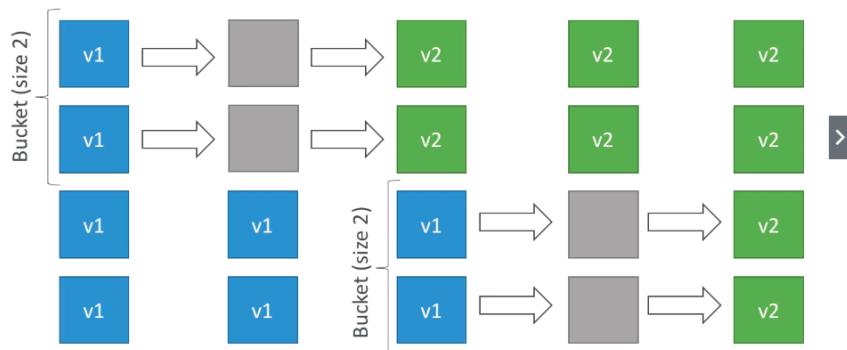


Und schließlich fallen bei diesem Setup keine zusätzlichen Kosten an.

Elastic Beanstalk Deployment

Rolling

- Application is running below capacity
- Can set the bucket size
- Application is running both versions simultaneously
- No additional cost
- Long deployment



Im Moment haben wir in diesem Beispiel eine

156. Beanstalk Deployment Modes

Elastic Beanstalk Deployment Rolling with additional batches

- Application is running at capacity
- Can set the bucket size
- Application is running both versions simultaneously
- Small additional cost
- Additional batch is removed at the end of the deployment
- Longer deployment
- Good for prod

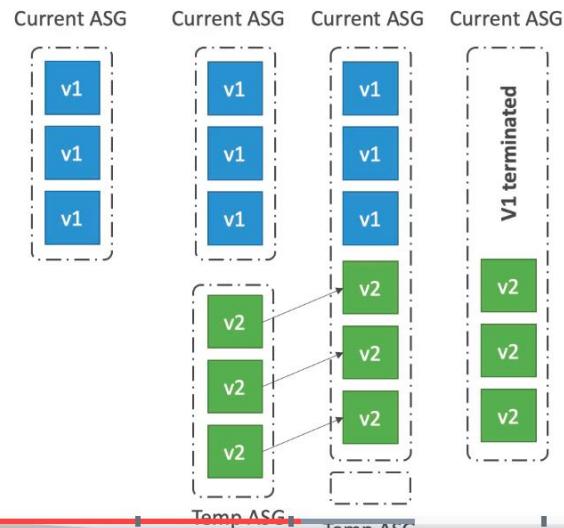


wird, ist zu jedem Zeitpunkt vier.

156. Beanstalk Deployment Modes

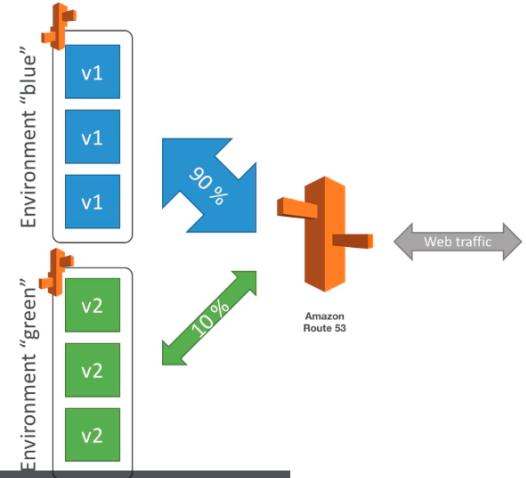
Elastic Beanstalk Deployment Immutable

- Zero downtime
- New Code is deployed to new instances on a temporary ASG
- High cost, double capacity
- Longest deployment
- Quick rollback in case of failures (just terminate new ASG)
- Great for prod



Elastic Beanstalk Deployment Blue / Green

- Not a “direct feature” of Elastic Beanstalk
- Zero downtime and release facility
- Create a new “stage” environment and deploy v2 there
- The new environment (green) can be validated independently and roll back if issues
- Route 53 can be setup using weighted policies to redirect a little bit of traffic to the stage environment
- Using Beanstalk, “swap URLs” when done with the environment test



AWS Certified Developer © Stephan Maarek

funktionieren, und vielleicht nur 10% des Datenverkehrs in

↓↓↓↓

156. Beanstalk Deployment Modes

Elastic Beanstalk Deployment Summary from AWS Doc

- <https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.deploy-existing-version.html>

Deployment Methods

Method	Impact of Failed Deployment	Deploy Time	Zero Downtime	No DNS Change	Rollback Process	Code Deployed To
All at once	Downtime	⌚	X	✓	Manual Redeploy	Existing instances
Rolling	Single batch out of service; any successful batches prior to failure running new application version	⌚⌚⌚	✓	✓	Manual Redeploy	Existing instances
Rolling with additional batch	Minimal if first batch fails, otherwise, similar to Rolling	⌚⌚⌚⌚	✓	✓	Manual Redeploy	New and existing instances
Immutable	Minimal	⌚⌚⌚⌚⌚	✓	✓	Terminate New Instances	New instances
Blue/green	Minimal	⌚⌚⌚⌚⌚⌚	✓	X	Swap URL	New instances

Anforderungen besser sind.

AWS Certified Developer © Stephan Maarek

↓↓↓↓

Elastic Beanstalk CLI

- We can install an additional CLI called the “EB cli” which makes working with Beanstalk from the CLI easier
- Basic commands are:
 - eb create
 - eb status
 - eb health
 - eb events
 - eb logs
 - eb open
 - eb deploy
 - eb config
 - eb terminate
- It's helpful for your automated deployment pipelines!

© Stephane Maarek

für die Entwicklerprüfung zu kennen.

by  [Stephane Maarek](#)

Elastic Beanstalk Deployment Process

- Describe dependencies
(requirements.txt for Python, package.json for Node.js)
 - Package code as zip, and describe dependencies
 - Python: requirements.txt
 - Node.js: package.json
 - Console: upload zip file (creates new app version), and then deploy
 - CLI: create new app version using CLI (uploads zip), and then deploy
-
- Elastic Beanstalk will deploy the zip on each EC2 instance, resolve dependencies and start the application

© Stephane Maarek

json for Node auf. js, und dann

by  [Stephane Maarek](#)

Beanstalk Lifecycle Policy

- Elastic Beanstalk can store at most 1000 application versions
- If you don't remove old versions, you won't be able to deploy anymore
- To phase out old application versions, use a **lifecycle policy**
 - Based on time (old versions are removed)
 - Based on space (when you have too many versions)
- Versions that are currently used won't be deleted
- Option not to delete the source bundle in S3 to prevent data loss

The screenshot shows a presentation slide with the following details:

- Top Bar:** "Schauen wir uns also an, wie das funktioniert." (Let's take a look at how it works.)
- Author:** © Stephane Maarek
- License:** CC BY-SA
- Section:** 160. Beanstalk Extensions
- Title:** Elastic Beanstalk Extensions
- Content:**
 - A zip file containing our code must be deployed to Elastic Beanstalk
 - All the parameters set in the UI can be configured with code using files
 - Requirements:
 - in the .ebextensions/ directory in the root of source code
 - YAML / JSON format
 - .config extensions (example: logging.config)
 - Able to modify some default settings using: option_settings
 - Ability to add resources such as RDS, ElastiCache, DynamoDB, etc...
 - Resources managed by .ebextensions get deleted if the environment goes away
- Bottom Bar:** "als Teil Ihrer Elastic"
- Video Player Controls:** Includes icons for volume, full screen, and other media controls.

Elastic Beanstalk Under the Hood

- Under the hood, Elastic Beanstalk relies on CloudFormation
- CloudFormation is used to provision other AWS services (we'll see later)



- Use case: you can define CloudFormation resources in your .ebextensions to provision ElastiCache, an S3 bucket, anything you want!

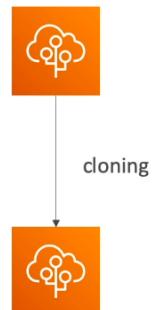
© Stephane Maarek

alles konfigurieren kann, was Sie in Ihrem AWS wollen.

by Sunny

Elastic Beanstalk Cloning

- Clone an environment with the exact same configuration
- Useful for deploying a “test” version of your application
- All resources and configuration are preserved:
 - Load Balancer type and configuration
 - RDS database type (but the data is not preserved)
 - Environment variables
- After cloning an environment, you can change settings



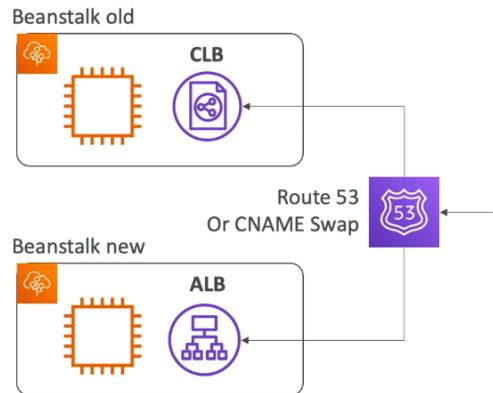
© Stephane Maarek

Umgebung können Sie deren Einstellungen ändern.

by Sunny

Elastic Beanstalk Migration: Load Balancer

- After creating an Elastic Beanstalk environment, you cannot change the Elastic Load Balancer type (only the configuration)
- To migrate:
 1. create a new environment with the same configuration except LB (can't clone)
 2. deploy your application onto the new environment
 3. perform a CNAME swap or Route 53 update



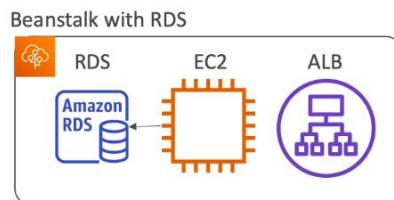
© Stephane Maarek

Route 53 verwenden, um ein DNS-Update durchzuführen. Dies

↳ [Route 53](#)

RDS with Elastic Beanstalk

- RDS can be provisioned with Beanstalk, which is great for dev / test
- This is not great for prod as the database lifecycle is tied to the Beanstalk environment lifecycle
- The best for prod is to separately create an RDS database and provide our EB application with the connection string



© Stephane Maarek

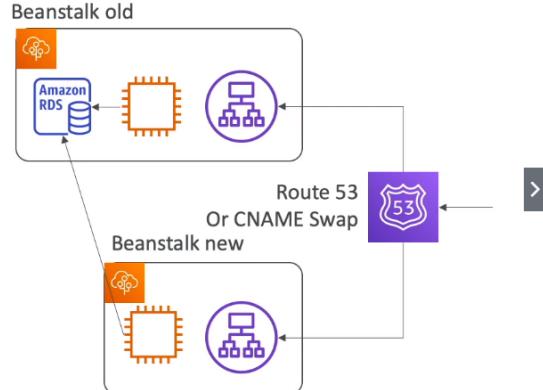
referenzieren, z. B. mithilfe einer Umgebungsvariablen.

↳ [Umgebungsvariablen](#)

163. Beanstalk Migrations

Elastic Beanstalk Migration: Decouple RDS

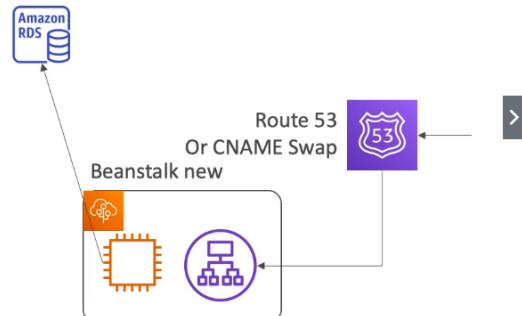
1. Create a snapshot of RDS DB (as a safeguard)
2. Go to the RDS console and protect the RDS database from deletion
3. Create a new Elastic Beanstalk environment, without RDS, point your application to existing RDS
4. perform a CNAME swap (blue/green) or Route 53 update, confirm working
5. Terminate the old environment (RDS won't be deleted)



163. Beanstalk Migrations

Elastic Beanstalk Migration: Decouple RDS

1. Create a snapshot of RDS DB (as a safeguard)
2. Go to the RDS console and protect the RDS database from deletion
3. Create a new Elastic Beanstalk environment, without RDS, point your application to existing RDS
4. perform a CNAME swap (blue/green) or Route 53 update, confirm working
5. Terminate the old environment (RDS won't be deleted)
6. Delete CloudFormation stack (in DELETE_FAILED state)



Wir müssen also einfach in

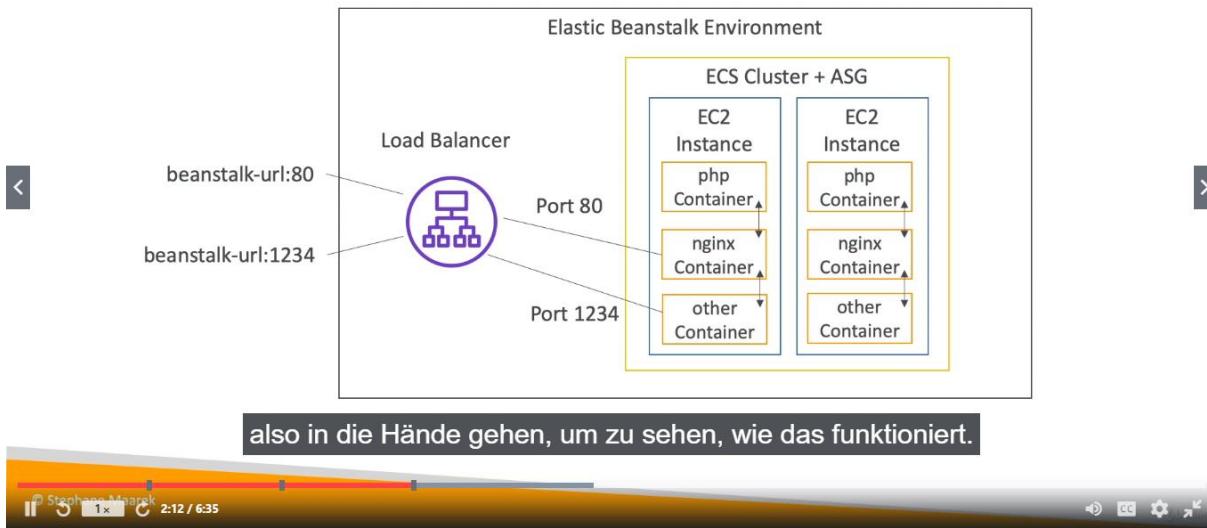
Elastic Beanstalk – Single Docker

- Run your application as a single docker container
- Either provide:
 - Dockerfile: Elastic Beanstalk will build and run the Docker container
 - Dockerrun.aws.json (v1): Describe where *already built* Docker image is
 - Image
 - Ports
 - Volumes
 - Logging
 - Etc...
- Beanstalk in Single Docker Container does not use ECS

The screenshot shows a presentation slide with the following details:

- Title:** Kulissen verwendet, sondern nur Docker auf EC2.
- Section:** 164. Beanstalk with Docker
- Section Header:** Elastic Beanstalk – Multi Docker Container
- Content:**
 - Multi Docker helps run multiple containers per EC2 instance in EB
 - This will create for you:
 - ECS Cluster
 - EC2 instances, configured to use the ECS Cluster
 - Load Balancer (in high availability mode)
 - Task definitions and execution
 - Requires a config **Dockerrun.aws.json (v2)** at the root of source code
 - **Dockerrun.aws.json** is used to generate the ECS task definition
 - Your Docker images must be pre-built and stored in ECR for example
- Text Overlay:** vorgefertigt und in einem Repository wie

Elastic Beanstalk + Multi Docker ECS



Elastic Beanstalk and HTTPS

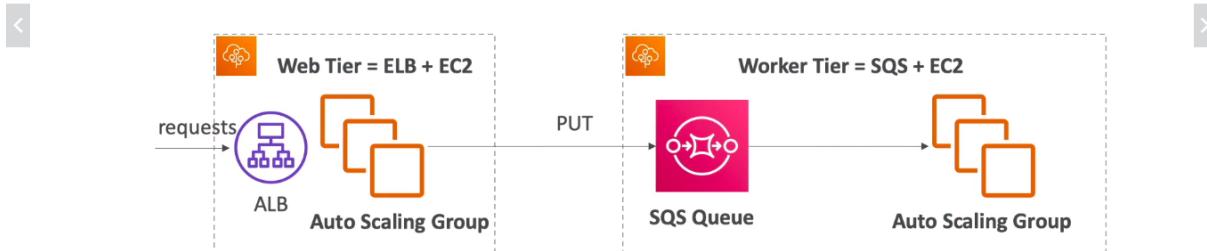
- Beanstalk with HTTPS
 - Idea: Load the SSL certificate onto the Load Balancer
 - Can be done from the Console (EB console, load balancer configuration)
 - Can be done from the code: .ebextensions/securelistener-alb.config
 - SSL Certificate can be provisioned using ACM (AWS Certificate Manager) or CLI
 - Must configure a security group rule to allow incoming port 443 (HTTPS port)

- Beanstalk redirect HTTP to HTTPS
 - Configure your instances to redirect HTTP to HTTPS:
<https://github.com/awsdocs/elastic-beanstalk-samples/tree/master/configuration-files/aws-provided/security-configuration/https-redirect>
 - OR configure the Application Load Balancer (ALB only) with a rule
 - Make sure health checks are not redirected (so they keep giving 200 OK)

damit Sie bei diesen Integritätsprüfungen weiterhin 200 OK erhalten .

Web Server vs Worker Environment

- If your application performs tasks that are long to complete, offload these tasks to a dedicated worker environment
- Decoupling your application into two tiers is common
- Example: processing a video, generating a zip file, etc
- You can define periodic tasks in a file cron.yaml



Elastic Beanstalk Custom Platform (Advanced)

- Custom Platforms are very advanced, they allow to define from scratch:
 - The Operating System (OS)
 - Additional Software
 - Scripts that Beanstalk runs on these platforms
- Use case: app language is incompatible with Beanstalk & doesn't use Docker
- To create your own platform:
 - Define an AMI using Platform.yaml file
 - Build that platform using the Packer software (open source tool to create AMIs)
- Custom Platform vs Custom Image (AMI):
 - Custom Image is to tweak an existing Beanstalk Platform (Python, Node.js, Java...)
 - Custom Platform is to create an entirely new Beanstalk Platform

Beanstalk für die Prüfung wissen sollten. Ich hoffe,

CICD Section Introduction

- We now know how to create resources in AWS manually (Fundamentals)
- We know how to interact with AWS programmatically (CLI)
- We've seen how to deploy code to AWS using Elastic Beanstalk
- All these manual steps make it very likely for us to do mistakes!
- What we'd like is to push our code "in a repository" and have it deployed onto the AWS
 - Automatically
 - The right way
 - Making sure it's tested before deploying
 - With possibility to go into different stages (dev, test, pre-prod, prod)
 - With manual approval where needed
- To be a proper AWS developer... we need to learn AWS CICD

AWS Certified Developer © Stephane Maarek

zu sein, benötigen wir AWS CICD.

168. Introduction to CICD in AWS

CICD Section Introduction

- This section is all about automating the deployment we've done so far while adding increased safety.
- It correspond to a whole part of the AWS Certification

< >

- We'll learn about
 - AWS CodeCommit: storing our code
 - AWS CodePipeline: automating our pipeline from code to ElasticBeanstalk
 - AWS CodeBuild: building and testing our code
 - AWS CodeDeploy: deploying the code to EC2 fleets (not Beanstalk)

AWS Certified Developer © Stephane Maarek

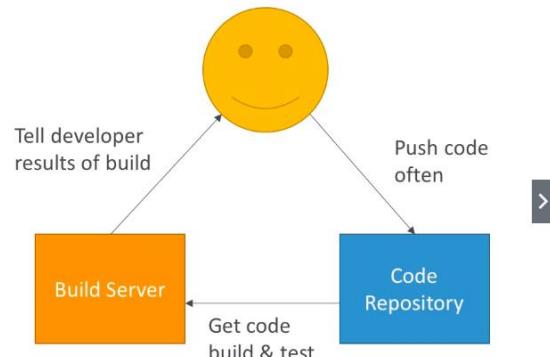
im Laufe

2.23 / 7:39

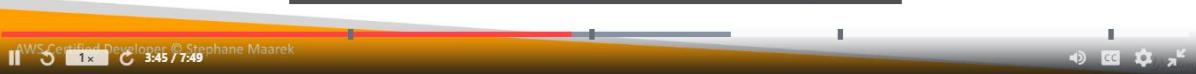
Speaker icon, Volume icon, CC icon, Settings icon, Fullscreen icon

Continuous Integration

- Developers push the code to a code repository often (GitHub / CodeCommit / Bitbucket / etc...)
- A testing / build server checks the code as soon as it's pushed (CodeBuild / Jenkins CI / etc...)
- The developer gets feedback about the tests and checks that have passed / failed
- Find bugs early, fix bugs
- Deliver faster as the code is tested
- Deploy often
- Happier developers, as they're unblocked

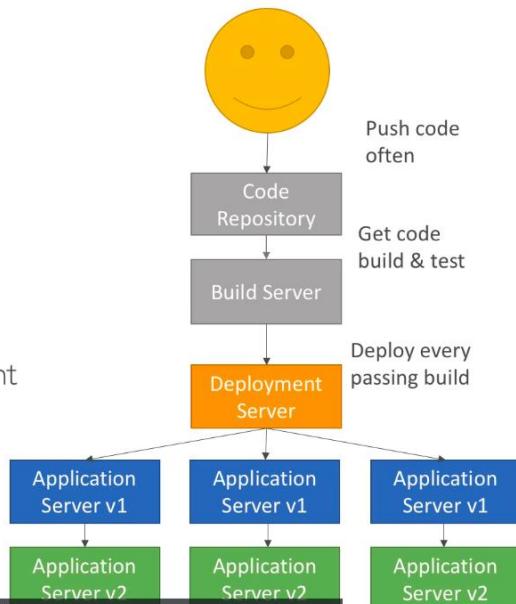


Und es soll Entwickler glücklicher machen, da sie

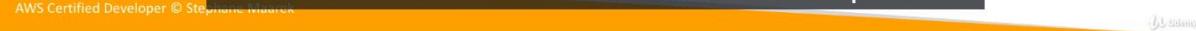


Continuous Delivery

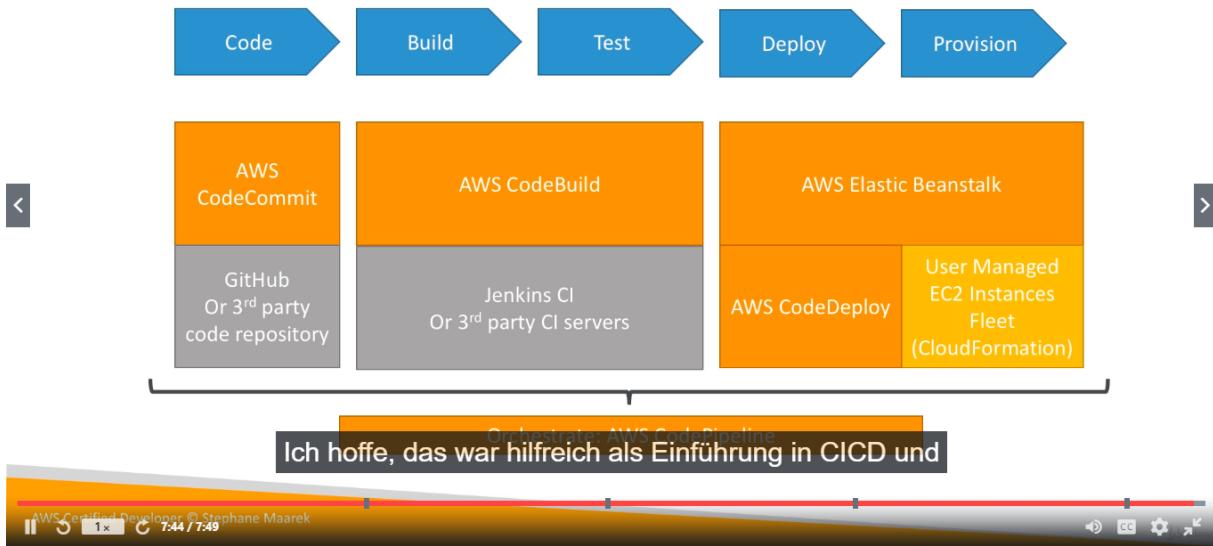
- Ensure that the software can be released reliably whenever needed.
- Ensures deployments happen often and are quick
- Shift away from "one release every 3 months" to "5 releases a day"
- That usually means automated deployment
 - CodeDeploy
 - Jenkins CD
 - Spinnaker
 - Etc...



Und das wird von Version zwei bis Version drei usw. passieren.



Technology Stack for CICD



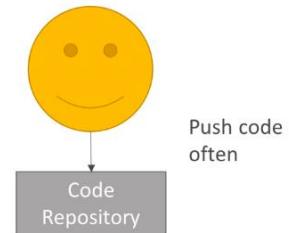
CodeCommit

- **Version control** is the ability to understand the various changes that happened to the code over time (and possibly roll back).
- All these are enabled by using a version control system such as Git
- A Git repository can live on one's machine, but it usually lives on a central online repository
- Benefits are:
 - Collaborate with other developers
 - Make sure the code is backed-up somewhere
 - Make sure it's fully viewable and auditable

beginnen, werden Sie wahrscheinlich nicht mehr davon loskommen.

CodeCommit

- Git repositories can be expensive.
- The industry includes:
 - GitHub: free public repositories, paid private ones
 - BitBucket
 - Etc...
- And AWS CodeCommit:
 - private Git repositories
 - No size limit on repositories (scale seamlessly)
 - Fully managed, highly available
 - Code only in AWS Cloud account => increased security and compliance
 - Secure (encrypted, access control, etc...)
 - Integrated with Jenkins / CodeBuild / other CI tools



AWS Certified Developer © Stephane Maarek

voll funktionsfähig und ziemlich sicher ist.

169. CodeCommit Overview

CodeCommit Security

< >

- Interactions are done using Git (standard)
- Authentication in Git:
 - SSH Keys: AWS Users can configure SSH keys in their IAM Console
 - HTTPS: Done through the AWS CLI Authentication helper or Generating HTTPS credentials
 - MFA (multi factor authentication) can be enabled for extra safety
- Authorization in Git:
 - IAM Policies manage user / roles rights to repositories
- Encryption:
 - Repositories are automatically encrypted at rest using KMS
 - Encrypted in transit (can only use HTTPS or SSH – both secure)
- Cross Account access:
 - Do not share your SSH keys
 - Do not share your AWS credentials
 - Use IAM Role in your AWS Account and use AWS STS (with AssumeRole API)

Sie müssen nicht genau wissen, wie

AWS Certified Developer © Stephane Maarek

5:34 / 9:02

CC Screenshot

CodeCommit vs GitHub

Similarities:

- Both are git repositories
 - Both support code review (pull requests)
 - GitHub and CodeCommit can be integrated with AWS CodeBuild
 - Both support HTTPS and SSH methods of authentication
- funktionsfähig ist. Die CodeCommit-Benutzeroberfläche ist ein bisschen minimal.**

Differences:

- Security:
 - GitHub: GitHub Users
 - CodeCommit: AWS IAM users & roles,
- Hosted:
 - GitHub: hosted by GitHub
 - GitHub Enterprise: self hosted on your servers
 - CodeCommit: managed & hosted by AWS
- UI:
 - GitHub UI is fully featured
 - CodeCommit UI is minimal

CodeCommit Notifications

- You can trigger notifications in CodeCommit using **AWS SNS (Simple Notification Service)** or **AWS Lambda** or **AWS CloudWatch Event Rules**
- Use cases for notifications SNS / AWS Lambda notifications:
 - Deletion of branches
 - Trigger for pushes that happens in master branch
 - Notify external Build System
 - Trigger AWS Lambda function to perform codebase analysis (maybe credentials got committed in the code?)
- Use cases for CloudWatch Event Rules:
 - Trigger for pull request updates (created / updated / deleted / commented)
 - Commit comment events
 - CloudWatch Event Rules goes into an SNS topic

Ich weiß, dass dies verwirrend sein kann, aber

CodePipeline



- Continuous delivery
- Visual workflow
- Source: GitHub / CodeCommit / Amazon S3
- Build: CodeBuild / Jenkins / etc...
- Load Testing: 3rd party tools
- Deploy: AWS CodeDeploy / Beanstalk / CloudFormation / ECS...
- Made of stages:
 - Each stage can have sequential actions and / or parallel actions
 - Stages examples: Build / Test / Deploy / Load Test / etc...
 - Manual approval can be defined at any stage

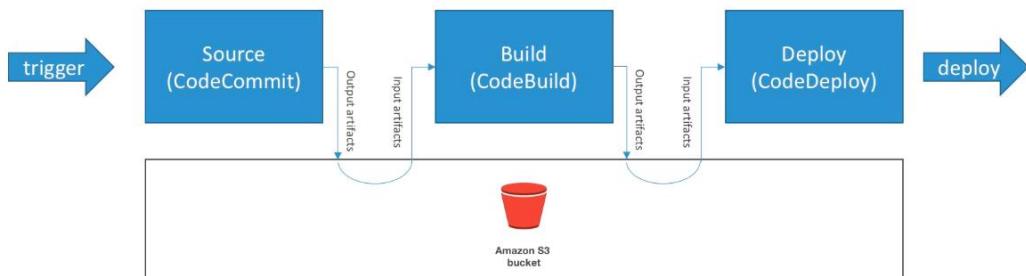
Es gibt eine Funktion, bei der Sie

AWS Certified Developer © Stephane Maarek

by Sunny

AWS CodePipeline Artifacts

- Each pipeline stage can create "artifacts"
- Artifacts are passed stored in Amazon S3 and passed on to the next stage



Dies ist jedoch nur eine vereinfachte Ansicht.

AWS Certified Developer © Stephane Maarek

by Sunny

CodePipeline Troubleshooting

- CodePipeline state changes happen in **AWS CloudWatch Events**, which can in return create SNS notifications.
 - Ex: you can create events for failed pipelines
 - Ex: you can create events for cancelled stages
- If CodePipeline fails a stage, your pipeline stops and you can get information in the console
- AWS CloudTrail can be used to audit AWS API calls
- If Pipeline can't perform an action, make sure the "IAM Service Role" attached does have enough permissions (IAM Policy)



174. CodeBuild Overview

CodeBuild Overview



- Fully managed build service
- Alternative to other build tools such as Jenkins
- Continuous scaling (no servers to manage or provision – no build queue)
- Pay for usage: the time it takes to complete the builds
- Leverages Docker under the hood for reproducible builds
- Possibility to extend capabilities leveraging our own base Docker images
- Secure: Integration with KMS for encryption of build artifacts, IAM for build permissions, and VPC for network security, CloudTrail for API calls logging



CodeBuild Overview



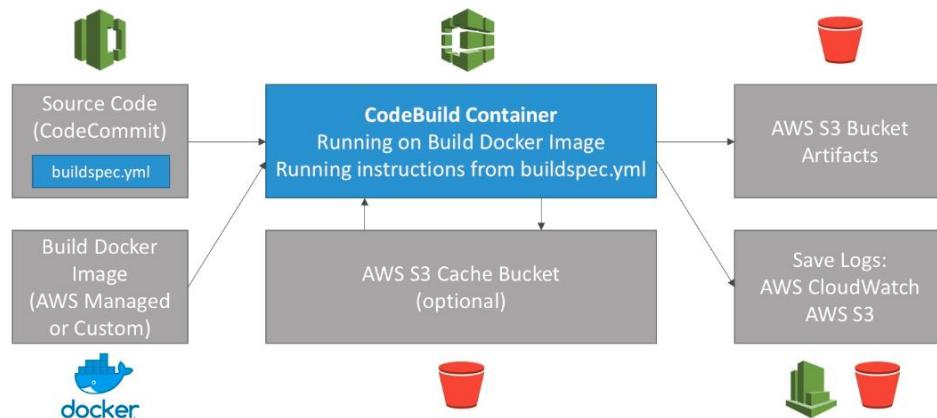
- Source Code from GitHub / CodeCommit / CodePipeline / S3...
- Build instructions can be defined in code (buildspec.yml file)
- Output logs to Amazon S3 & AWS CloudWatch Logs
- Metrics to monitor CodeBuild statistics
- Use CloudWatch Alarms to detect failed builds and trigger notifications
- CloudWatch Events / AWS Lambda as a Glue
- SNS notifications
- Ability to reproduce CodeBuild locally to troubleshoot in case of errors
- Builds can be defined within CodePipeline or CodeBuild itself
Unter Strich könnten wir also definieren, was wir für diese praktischen Übungen verwenden.

CodeBuild Support environments

- Java
- Ruby
- Python
- Go
- Node.js
- Android
- .NET Core
- PHP
- Docker: extend any environment you like

Und genau das gefällt mir sehr gut.

How CodeBuild works



Geschichte eingehen.

AWS Certified Developer © Stephane Maarek

Jörg Schmid

174. CodeBuild Overview

CodeBuild BuildSpec

- **buildspec.yml** file must be at the **root** of your code
- Define environment variables:
 - Plaintext variables
 - Secure secrets: use SSM Parameter store
- Phases (specify commands to run):
 - Install: install dependencies you may need for your build
 - Pre build: final commands to execute before build
 - Build: **actual build commands**
 - Post build: finishing touches (zip output for example)
- Artifacts: What to upload to S3 (encrypted with KMS)
- Cache: Files to cache (usually dependencies) to S3 for future build speedup

Das ist also eine Leistungsverbesserung.

AWS Certified Developer © Stephane Maarek

Jörg Schmid

CodeBuild Local Build

- In case of need of deep troubleshooting beyond logs...
- You can run CodeBuild locally on your desktop (after installing Docker)
- For this, leverage the CodeBuild Agent

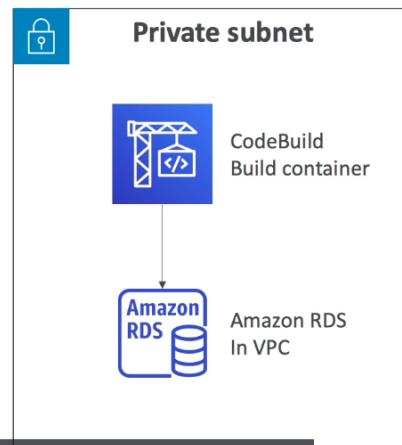
- < <https://docs.aws.amazon.com/codebuild/latest/userguide/use-codebuild-agent.html> >

wird und einen Agenten erstellen kann, der ziemlich fortgeschritten ist, aber Wenn Sie interessiert sind, gehen Sie einfach auf Google Code



CodeBuild in VPC

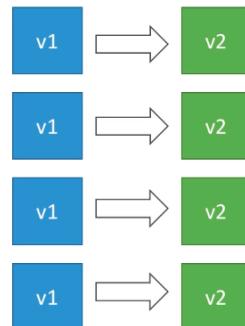
- By default, your CodeBuild containers are launched outside your VPC
- Therefore, by default it cannot access resources in a VPC
- You can specify a VPC configuration:
 - VPC ID
 - Subnet IDs
 - Security Group IDs
- Then your build can access resources in your VPC (RDS, ElastiCache, EC2, ALB..)
- Use cases: integration tests, data query, internal load balancers



ausführen oder mit Ihren internen Load Balancern interagieren möchten, da

AWS CodeDeploy

- We want to deploy our application automatically to many EC2 instances
- These instances are not managed by Elastic Beanstalk
- There are several ways to handle deployments using open source tools (Ansible, Terraform, Chef, Puppet, etc...)
- We can use the managed Service AWS CodeDeploy



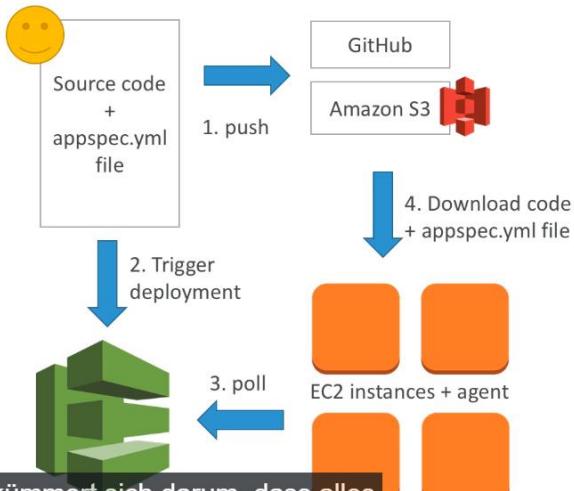
AWS Certified Developer © Stephane Maarek

da es sich um einen verwalteten Dienst handelt und

JB Savary

AWS CodeDeploy – Steps to make it work

- Each EC2 Machine (or On Premise machine) must be running the CodeDeploy Agent
- The agent is continuously polling AWS CodeDeploy for work to do
- CodeDeploy sends **appspec.yml** file.
- Application is pulled from GitHub or S3
- EC2 will run the deployment instructions
- CodeDeploy Agent will report of success / failure of deployment on the instance



AWS Certified Developer © Stephane Maarek

EC2-Instanzen und der Agent kümmert sich darum, dass alles

JB Savary

AWS CodeDeploy – Other

- EC2 instances are grouped by deployment group (dev / test / prod)
- Lots of flexibility to define any kind of deployments
- CodeDeploy can be chained into CodePipeline and use artifacts from there
- CodeDeploy can re-use existing setup tools, works with any application, auto scaling integration
- Note: Blue / Green only works with EC2 instances (not on premise)
- Support for AWS Lambda deployments (we'll see this later)
- CodeDeploy does not provision resources

wissen, dass CodeDeploy nur Ihre Anwendung bereitstellt.



AWS CodeDeploy Primary Components

- **Application:** unique name
- **Compute platform:** EC2/On-Premise or Lambda
- **Deployment configuration:** Deployment rules for success / failures
 - EC2/On-Premise: you can specify the minimum number of healthy instances for the deployment.
 - AWS Lambda: specify how traffic is routed to your updated Lambda function versions.
- **Deployment group:** group of tagged instances (allows to deploy gradually)
- **Deployment type:** In-place deployment or Blue/green deployment:
- **IAM instance profile:** need to give EC2 the permissions to pull from S3 / GitHub
- **Application Revision:** application code + appspec.yml file
- **Service role:** Role for CodeDeploy to perform what it needs
- **Target revision:** Target deployment application version

lernen, sondern nur allgemein, um eine Vorstellung davon zu bekommen, was sie sind.

178. CodeDeploy Overview

AWS CodeDeploy AppSpec

- File section: how to source and copy from S3 / GitHub to filesystem
- Hooks: set of instructions to do to deploy the new version (hooks can have timeouts). The order is:
 - ApplicationStop
 - DownloadBundle
 - BeforeInstall
 - AfterInstall
 - ApplicationStart
 - ValidateService: really important



zum Beispiel nach der Reihenfolge dieser Haken.



178. CodeDeploy Overview

AWS CodeDeploy Deploy & Hooks Order

Event	Start time	End time	Duration	Status
ApplicationStop	Sept 26, 2018 7:51:29 AM UTC	Sept 26, 2018 7:51:29 AM UTC	less than one second	Succeeded
DownloadBundle	Sept 26, 2018 7:51:30 AM UTC	Sept 26, 2018 7:51:30 AM UTC	less than one second	Succeeded
BeforeInstall	Sept 26, 2018 7:51:31 AM UTC	Sept 26, 2018 7:51:32 AM UTC	2 secs	Succeeded
Install	Sept 26, 2018 7:51:33 AM UTC	Sept 26, 2018 7:51:33 AM UTC	less than one second	Succeeded
AfterInstall	Sept 26, 2018 7:51:34 AM UTC	Sept 26, 2018 7:51:34 AM UTC	less than one second	Succeeded
ApplicationStart	Sept 26, 2018 7:51:35 AM UTC	Sept 26, 2018 7:51:35 AM UTC	less than one second	Succeeded
ValidateService	Sept 26, 2018 7:51:36 AM UTC	Sept 26, 2018 7:51:36 AM UTC	less than one second	Succeeded
BeforeAllowTraffic	Sept 26, 2018 7:51:49 AM UTC	Sept 26, 2018 7:51:49 AM UTC	less than one second	Succeeded
AllowTraffic	Sept 26, 2018 7:51:50 AM UTC	Sept 26, 2018 7:52:11 AM UTC	21 secs	Succeeded
AfterAllowTraffic	Sept 26, 2018 7:52:12 AM UTC	Sept 26, 2018 7:52:12 AM UTC	less than one second	Succeeded



haben im Grunde die Kontrolle über viele von ihnen.



AWS CodeDeploy Deployment Config

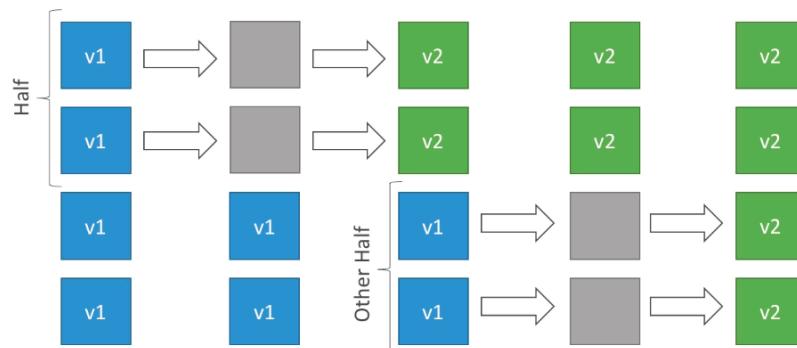
- Configs:
 - One at a time: one instance at a time, one instance fails => deployment stops
 - Half at a time: 50%
 - All at once: quick but no healthy host, downtime. Good for dev
 - Custom: min healthy host = 75%
- Failures:
 - Instances stay in “failed state”
 - New deployments will first be deployed to “failed state” instances
 - To rollback: redeploy old deployment or enable automated rollback for failures
- Deployment Targets:
 - Set of EC2 instances with tags
 - Directly to an ASG
 - Mix of ASG / Tags so you can build deployment segments
 - Customization in scripts with DEPLOYMENT_GROUP_NAME environment variables

es sich bei einem Bereitstellungsziel um EC2-Instanzen mit Tags oder eine

AWS Certified Developer © Stephanie Mäurer

by  [Stephanie Mäurer](#)

In Place Deployment – Half at a time

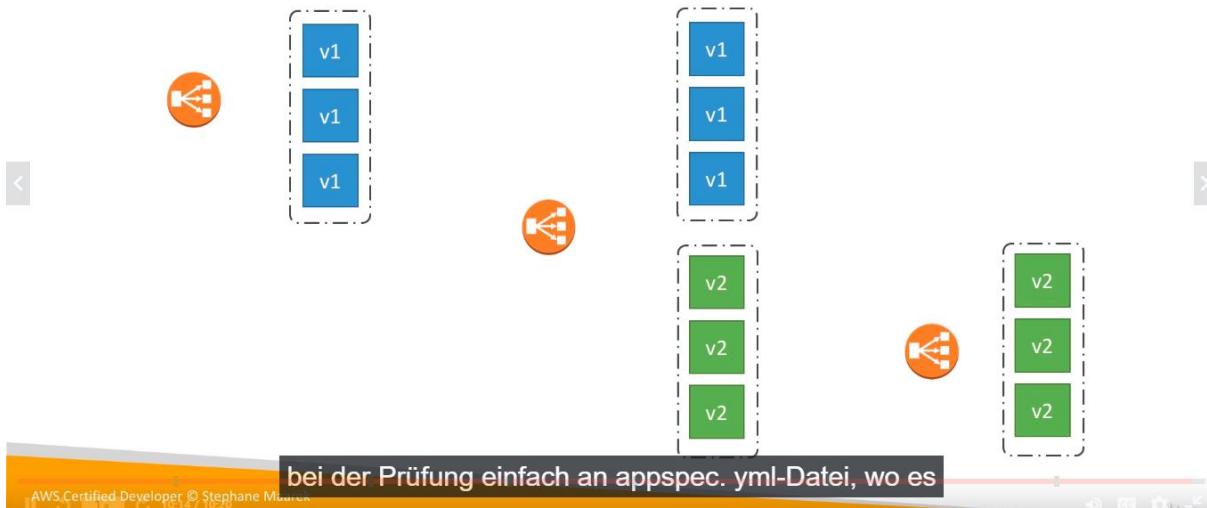


da ich denke, Elastic Beanstalk verwendet CodeDeploy unter der Motorhaube, aber Sie sehen es nicht.

AWS Certified Developer © Stephanie Mäurer

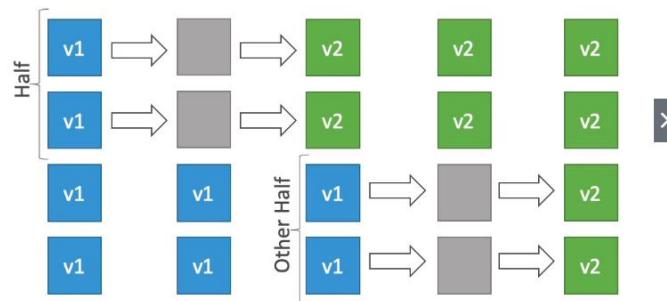
by  [Stephanie Mäurer](#)

Blue Green Deployment



CodeDeploy to EC2

- Define how to deploy the application using appspec.yml + deployment strategy
- Will do in-place update to your fleet of EC2 instances
- Can use hooks to verify the deployment after each deployment phase

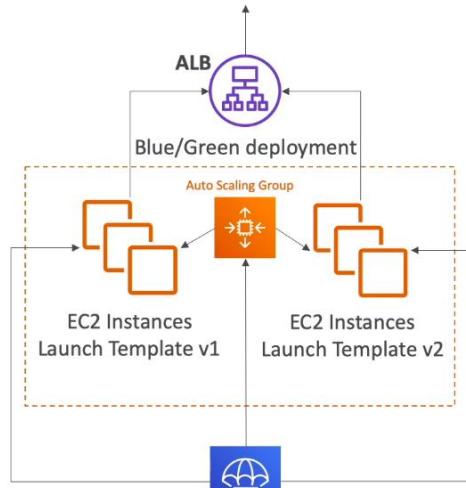


Version zwei aktualisiert.



CodeDeploy to ASG

- In place updates:
 - Updates current existing EC2 instances
 - Instances newly created by an ASG will also get automated deployments
- Blue / green deployment:
 - A new auto-scaling group is created (settings are copied)
 - Choose how long to keep the old instances
 - Must be using an ELB



vorherigen Instanzen aufbewahrt werden sollen. Wenn diese Zeit
abgelaufen ist, werden die

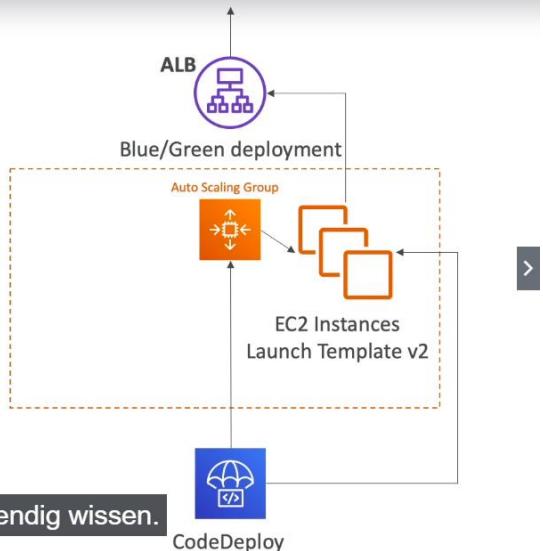
© Stephane Maarek

CC BY-NC

180. CodeDeploy for EC2 and ASG

CodeDeploy to ASG

- In place updates:
 - Updates current existing EC2 instances
 - Instances newly created by an ASG will also get automated deployments
- Blue / green deployment:
 - A new auto-scaling group is created (settings are copied)
 - Choose how long to keep the old instances
 - Must be using an ELB



müssen das nicht auswendig wissen.

CodeDeploy

© Stephane Maarek 1x 2:25 / 3:29

CC BY-NC

CodeDeploy - roll backs

- You can specify automated rollback options
- Roll back when a deployment fails
- Roll back when alarm thresholds are met
- Disable rollbacks — Do not perform rollbacks for this deployment.
- If a roll back happens, CodeDeploy redeploys the last known good revision as a new deployment.

The screenshot shows a presentation slide with the following elements:

- Header:** als neue Bereitstellung erneut bereit.
- Navigation:** © Stephane Maarek, **181. CodeStar - Overview**, **CodeStar**, **Stephane Maarek**, **zum Video**.
- Content:** A bulleted list describing the features of AWS CodeStar:
 - CodeStar is an integrated solution that regroups: GitHub, CodeCommit, CodeBuild, CodeDeploy, CloudFormation, CodePipeline, CloudWatch
 - Helps quickly create “CICD-ready” projects for EC2, Lambda, Beanstalk
 - Supported languages: C#, Go, HTML 5, Java, Node.js, PHP, Python, Ruby
 - Issue tracking integration with: JIRA / GitHub Issues
 - Ability to integrate with Cloud9 to obtain a web IDE (not all regions)
 - One dashboard to view all your components
 - Free service, pay only for the underlying usage of other services
 - Limited Customization
- Footer:** zugrunde liegenden Dienstes bearbeiten können. AWS Certified Developer © Stephane Maarek.

Infrastructure as Code

- Currently, we have been doing a lot of manual work
- All this manual work will be very tough to reproduce:
 - In another region
 - in another AWS account
 - Within the same region if everything was deleted
- Wouldn't it be great, if all our infrastructure was... code?
- That code would be deployed and create / update / delete our infrastructure

The screenshot shows a presentation slide with the following elements:

- Title Bar:** "Code, den wir schreiben werden, bereitstellen können und der" (Code we will write, can be provided and) and "AWS Certified Developer © Stephane Maarek".
- Navigation:** "184. CloudFormation Overview" on the left, and navigation arrows (< and >) on the right.
- Section Header:** "What is CloudFormation"
- Image:** A green 3D cube icon representing CloudFormation.
- Text:** A bulleted list describing CloudFormation's declarative nature and its ability to create complex infrastructure from simple templates.
- Text Box:** "ist irgendwie nett." (is somehow nice.)
- Player Controls:** Standard video player controls at the bottom, including volume, brightness, and playback progress (1:54 / 7:06).

Benefits of AWS CloudFormation (1/2)

- Infrastructure as code
 - No resources are manually created, which is excellent for control
 - The code can be version controlled for example using git
 - Changes to the infrastructure are reviewed through code
- Cost
 - Each resources within the stack is tagged with an identifier so you can easily see how much a stack costs you
 - You can estimate the costs of your resources using the CloudFormation template
 - Savings strategy: In Dev, you could automation deletion of templates at 5 PM and recreated at 8 AM, safely

AWS Certified Developer © Stephane Maarek

Ihrem kleinen AWS-Konto haben möchten, können Sie

184. CloudFormation Overview

Benefits of AWS CloudFormation (2/2)

- Productivity
 - Ability to destroy and re-create an infrastructure on the cloud on the fly
 - Automated generation of Diagram for your templates!
 - Declarative programming (no need to figure out ordering and orchestration)
- Separation of concern: create many stacks for many apps, and many layers. Ex:
 - VPC stacks
 - Network stacks
 - App stacks
- Don't re-invent the wheel
 - Leverage existing templates on the web!
 - Leverage the documentation

finden Sie alles, was Sie

AWS Certified Developer © Stephane Maarek

3:54 / 7:06

CC

How CloudFormation Works

- Templates have to be uploaded in S3 and then referenced in CloudFormation
- To update a template, we can't edit previous ones. We have to re-upload a new version of the template to AWS
- Stacks are identified by a name
- Deleting a stack deletes every single artifact that was created by CloudFormation.

erstellt wurden, löschen können und sicher sind, dass Sie nichts zurücklassen.



Deploying CloudFormation templates

- Manual way:
 - Editing templates in the CloudFormation Designer
 - Using the console to input parameters, etc
- Automated way:
 - Editing templates in a YAML file
 - Using the AWS CLI (Command Line Interface) to deploy the templates
 - Recommended way when you fully want to automate your flow

entweder manuell oder automatisiert wählen. Ich denke,

CloudFormation Building Blocks

Templates components (one course section for each):

1. Resources: your AWS resources declared in the template (MANDATORY)
2. Parameters: the dynamic inputs for your template
3. Mappings: the static variables for your template
4. Outputs: References to what has been created
5. Conditionals: List of conditions to perform resource creation
6. Metadata

Templates helpers:

1. References
2. Functions

Funktionen zum Transformieren von Daten in Ihren Vorlagen verwenden können.

AWS Certified Developer © Stephane Maarek

by Sammy

YAML Crash Course

```
1 invoice:      34843
2 date :        2001-01-23
3 bill-to:
4   given :  Chris
5   family : Dumars
6   address:
7     lines: |
8       458 Walkman Dr.
9       Suite #292
10  city  : Royal Oak
11  state : MI
12  postal : 48046
13 product:
14   - sku      : BL394D
15   quantity  : 4
16   description : Basketball
17   price     : 450.00
18   - sku      : BL4438H
19   quantity  : 1
20   description : Super Hoop
21   price     : 2392.00
```

- YAML and JSON are the languages you can use for CloudFormation.
- JSON is horrible for CF
- YAML is great in so many ways
- Let's learn a bit about it!
- Key value Pairs
- Nested objects
- Support Arrays
- Multi line strings
- Can include comments!

haben wir die Idee.

AWS Certified Developer © Stephane Maarek

by Sammy

What are resources?

- Resources are the core of your CloudFormation template (MANDATORY)
- They represent the different AWS Components that will be created and configured
- Resources are declared and can reference each other



- AWS figures out creation, updates and deletes of resources for us
- There are over 224 types of resources (!)
- Resource types identifiers are of the form:

AWS::aws-product-name::data-type-name

Normalerweise lesen Sie diese Bezeichnervariable, um



How do I find resources documentation?

- I can't teach you all of the 224 resources, but I can teach you how to learn how to use them.
- All the resources can be found here:
<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html>
- Then, we just read the docs ☺
- Example here (for an EC2 instance):
<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-instance.html>



für die EC2-Instanz zusammen lesen, um ein



The screenshot shows the AWS CloudFormation User Guide (API Version 2010-05-15) with the title '188. CloudFormation Resources'. The main content is the 'AWS Resource Types Reference' section. It includes a search bar, a navigation bar with links like 'AWS Documentation', 'AWS CloudFormation', 'User Guide', 'Template Reference', and 'AWS Resource Types Reference'. A sidebar on the left lists topics such as 'Continuous Delivery', 'Working with Stacks', 'Working with Templates', 'Working with AWS CloudFormation StackSets', 'Template Reference', and 'AWS Resource Types' (which is expanded to show 'AmazonMQ::Broker', 'AmazonMQ::Configuration', etc.). A red box highlights the text 'Die Namenskonvention für Ressourcen ist hier und wenn' (The naming convention for resources is here and when). The bottom of the page shows standard browser controls and a footer with copyright information.

188. CloudFormation Resources

FAQ for resources

- Can I create a dynamic amount of resources?

➤ No, you can't. Everything in the CloudFormation template has to be declared. You can't perform code generation there

- Is every AWS Service supported?

➤ Almost. Only a select few niches are not there yet

➤ You can work around that using AWS Lambda Custom Resources

Hoffe das macht jetzt mehr Sinn



Parameters Settings

Parameters can be controlled by all these settings:

- **Type:**
 - String
 - Number
 - CommaDelimitedList
 - List<Type>
 - AWS Parameter (to help catch invalid values – match against existing values in the AWS Account)
- **Description**
- **Constraints**
 - ConstraintDescription (String)
 - Min/MaxLength
 - Min/MaxValue
 - Defaults
 - AllowedValues (array)
 - AllowedPattern (regexp)
 - NoEcho (Boolean)

was Sie wissen sollten, denke ich für die Prüfung.

AWS Certified Developer © Stephane Maarek

Überprüfen

Concept: Pseudo Parameters

- AWS offers us pseudo parameters in any CloudFormation template.
- These can be used at any time and are enabled by default

Reference Value	Example Return Value
AWS::AccountId	1234567890
AWS::NotificationARNs	[arn:aws:sns:us-east-1:123456789012:MyTopic]
AWS::NoValue	Does not return a value.
AWS::Region	us-east-2
AWS::StackId	arn:aws:cloudformation:us-east-1:123456789012:stack/MyStack/1c2fa620-982a-11e3-aff7-50e2416294e0
AWS::StackName	MyStack

nicht zurückgeben möchten.

AWS Certified Developer © Stephane Maarek

Überprüfen

What are mappings?

- Mappings are fixed variables within your CloudFormation Template.
- They're very handy to differentiate between different environments (dev vs prod), regions (AWS regions), AMI types, etc
- All the values are hardcoded within the template
- Example:

```

Mappings:
  Mapping01:
    Key01:
      Name: Value01
    Key02:
      Name: Value02
    Key03:
      Name: Value03

RegionMap:
  us-east-1:
    "32": "ami-6411e20d"
    "64": "ami-7a11e213"
  us-west-1:
    "32": "ami-c9c7978c"
    "64": "ami-cfc7978a"
    "64": "ami-31c2f645"

```

Hier ist die AMI-ID, die Sie verwenden sollten.

When would you use mappings vs parameters ?

- Mappings are great when you know in advance all the values that can be taken and that they can be deduced from variables such as
 - Region
 - Availability Zone
 - AWS Account
 - Environment (dev vs prod)
 - Etc...
- They allow safer control over the template.
- Use parameters when the values are really user specific



Fn::FindInMap

Accessing Mapping Values

- We use **Fn::FindInMap** to return a named value from a specific key
- **!FindInMap [MapName, TopLevelKey, SecondLevelKey]**

```

AWSTemplateFormatVersion: "2010-09-09"
Mappings:
  RegionMap:
    us-east-1:
      "32": "ami-6411e20d"
      "64": "ami-7a11e213"
    us-west-1:
      "32": "ami-c9c7978c"
      "64": "ami-cfc7978a"
    eu-west-1:
      "32": "ami-37c2f643"
      "64": "ami-31c2f645"
    ap-southeast-1:
      "32": "ami-16ff28c34"
      "64": "ami-16ff28c32"
    ap-northeast-1:
      "32": "ami-9c03a89d"
      "64": "ami-a003a8a1"
Resources:
  myEC2Instance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: !FindInMap [RegionMap, !Ref "AWS::Region", 32]
      InstanceType: mi.small
  
```

AWS Certified Developer © Stephanie Maarek 2:41 / 2:53

What are outputs?

- The Outputs section declares *optional* outputs values that we can import into other stacks (if you export them first)!
- You can also view the outputs in the AWS Console or in using the AWS CLI
- They're very useful for example if you define a network CloudFormation, and output the variables such as VPC ID and your Subnet IDs
- It's the best way to perform some collaboration cross stack, as you let expert handle their own part of the stack
- You can't delete a CloudFormation Stack if its outputs are being referenced by another CloudFormation stack

diese von einem anderen CloudFormation-Stapel referenziert werden.

191. CloudFormation Outputs

Outputs Example

- Creating a SSH Security Group as part of one template
- We create an output that references that security group

Outputs:

StackSSHSecurityGroup:

Description: The SSH Security Group for our Company

Value: !Ref MyCompanyWideSSHSecurityGroup

Export:

Name: SSHSecurityGroup

Die Syntax ist also

191. CloudFormation Outputs

Cross Stack Reference

- We then create a second template that leverages that security group
- For this, we use the **Fn::ImportValue** function
- You can't delete the underlying stack until all the references are deleted too.

Resources:

MySecureInstance:

Type: AWS::EC2::Instance

Properties:

AvailabilityZone: us-east-1a

ImageId: ami-a4c7edb2

InstanceType: t2.micro

SecurityGroups:

- !ImportValue SSHSecurityGroup

Importwert gibt, und dann verweisen wir auf genau denselben

192. CloudFormation Conditions

What are conditions used for?

- Conditions are used to control the creation of resources or outputs based on a condition.
- Conditions can be whatever you want them to be, but common ones are:
 - Environment (dev / test / prod)
 - AWS Region
 - Any parameter value
- Each condition can reference another condition, parameter value or mapping

Die Zuordnung verweisen, sodass Sie sie einstellen können.

192. CloudFormation Conditions

How to define a condition?

Conditions:

```
| CreateProdResources: !Equals [ !Ref EnvType, prod ]
```

- The logical ID is for you to choose. It's how you name condition
- The intrinsic function (logical) can be any of the following:
 - Fn::And
 - Fn::Equals
 - Fn::If
 - Fn::Not
 - Fn::Or

Dies sind logische Funktionen,

192. CloudFormation Conditions

Using a Condition

- Conditions can be applied to resources / outputs / etc...

```
Resources:  
  MountPoint:  
    Type: "AWS::EC2::VolumeAttachment"  
    Condition: CreateProdResources
```

The screenshot shows a presentation slide with the following content:

AWS Certified Developer © Stephane Maarek

Sie befinden sich im Grunde genommen auf derselben

193. CloudFormation Intrinsic Functions

CloudFormation Must Know Intrinsic Functions

< >

- Ref
- Fn::GetAtt
- Fn::FindInMap
- Fn::ImportValue
- Fn::Join
- Fn::Sub
- Condition Functions (Fn::If, Fn::Not, Fn::Equals, etc...)

Auffrischung des Gedächtnisses darüber vornehmen, was sie sind.

AWS Certified Developer © Stephane Maarek

Fn::Ref

- The `Fn::Ref` function can be leveraged to reference
 - Parameters => returns the value of the parameter
 - Resources => returns the physical ID of the underlying resource (ex: EC2 ID)
- The shorthand for this in YAML is `!Ref`

```
DbSubnet1:  
  Type: AWS::EC2::Subnet  
  Properties:  
    VpcId: !Ref MyVPC
```

Wie gesagt, wenn Sie auf eine Ressource verweisen,

AWS Certified Developer © Stephane Maarek

Überblick

Fn::GetAtt

- Attributes are attached to any resources you create
- To know the attributes of your resources, the best place to look at is the documentation.
- For example: the AZ of an EC2 machine!

```
Resources:  
  EC2Instance:  
    Type: "AWS::EC2::Instance"  
    Properties:  
      ImageId: ami-1234567  
      InstanceType: t2.micro
```

```
NewVolume:  
  Type: "AWS::EC2::Volume"  
  Condition: CreateProdResources  
  Properties:  
    Size: 100  
    AvailabilityZone:  
      !GetAtt EC2Instance.AvailabilityZone
```

und wir werden sagen, dass der Typ ein EC2-Volume ist.

AWS Certified Developer © Stephane Maarek

Überblick

Function Fn::Sub

- Fn::Sub, or !Sub as a shorthand, is used to substitute variables from a text. It's a very handy function that will allow you to fully customize your templates.
- For example, you can combine Fn::Sub with References or AWS Pseudo variables!
- String must contain \${VariableName} and will substitute them

```
!Sub
- String
- { Var1Name: Var1Value, Var2Name: Var2Value }
```

```
!Sub String
```

Sub soll Werte ersetzen.

AWS Certified Developer © Stephane Maarek

by Sammy

CloudFormation Rollbacks

- Stack Creation Fails:
 - Default: everything rolls back (gets deleted). We can look at the log
 - Option to disable rollback and troubleshoot what happened
- Stack Update Fails:
 - The stack automatically rolls back to the previous known working state
 - Ability to see in the log what happened and error messages

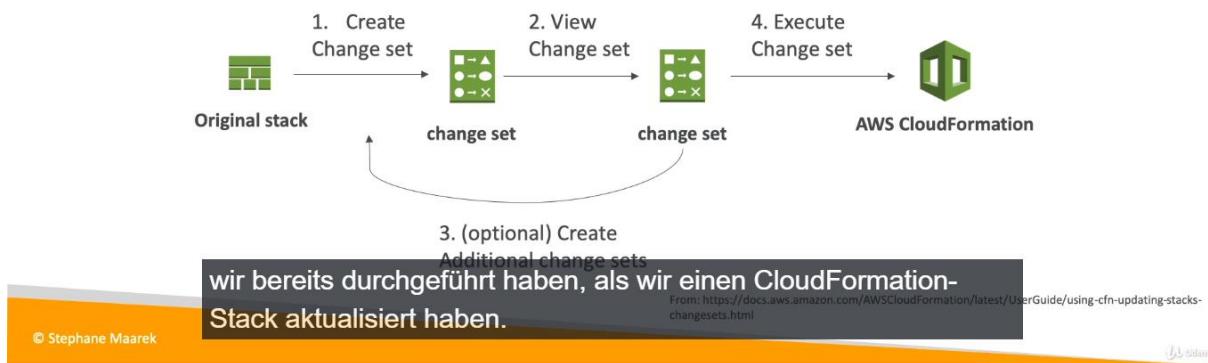
In diesem Vortrag erfahren Sie, wie Rollback funktioniert.

AWS Certified Developer © Stephane Maarek

by Sammy

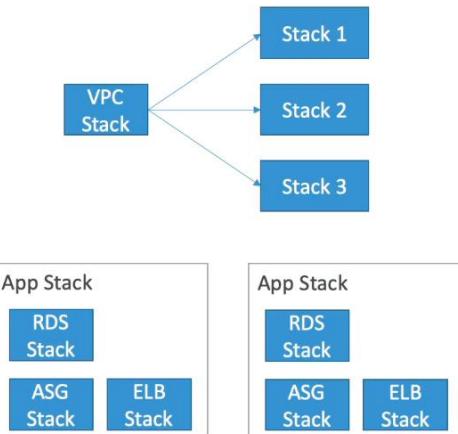
ChangeSets

- When you update a stack, you need to know what changes before it happens for greater confidence
- ChangeSets won't say if the update will be successful



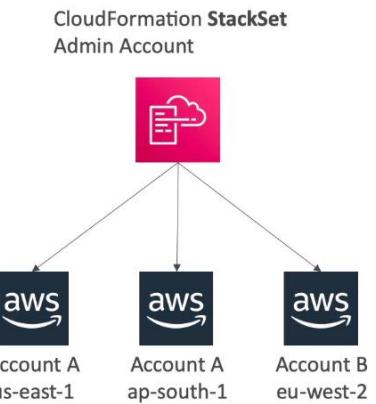
CloudFormation – Cross vs Nested Stacks

- Cross Stacks
 - Helpful when stacks have different lifecycles
 - Use Outputs Export and Fn::ImportValue
 - When you need to pass export values to many stacks (VPC Id, etc...)
- Nested Stacks
 - Helpful when components must be re-used
 - Ex: re-use how to properly configure an Application Load Balancer
 - The nested stack only is important to the higher level stack (it's not shared)



CloudFormation - StackSets

- Create, update, or delete stacks across multiple accounts and regions with a single operation
- Administrator account to create StackSets
- Trusted accounts to create, update, delete stack instances from StackSets
- When you update a stack set, *all* associated stack instances are updated throughout all accounts and regions.



© Stephane Maarek

hilfreich und wir sehen uns in der nächsten Vorlesung.

↓ Summary

197. Monitoring Overview in AWS

Why Monitoring is Important

- We know how to deploy applications
 - Safely
 - Automatically
 - Using Infrastructure as Code
 - Leveraging the best AWS components!
- Our applications are deployed, and our users don't care how we did it...
- Our users only care that the application is working!
 - Application latency: will it increase over time?
 - Application outages: customer experience should not be degraded
 - Users contacting the IT department or complaining is not a good outcome
 - Troubleshooting and remediation
- Internal monitoring:
 - Can we prevent issues before they happen?
 - Performance and Cost
 - Trends (scaling patterns)
 - Learning and Improvement

Für mich ist Überwachung also wirklich sehr wichtig.

AWS Certified Developer © Stephane Maarek

◀ ▶ 🔍 CC ⚙️ ⌂

1x 1:31 / 2:44

Monitoring in AWS

- AWS CloudWatch:
 - Metrics: Collect and track key metrics
 - Logs: Collect, monitor, analyze and store log files
 - Events: Send notifications when certain events happen in your AWS
 - Alarms: React in real-time to metrics / events
- AWS X-Ray:
 - Troubleshooting application performance and errors
 - Distributed tracing of microservices
- AWS CloudTrail:
 - Internal monitoring of API calls being made
 - Audit changes to AWS Resources by your users

Insgesamt bieten diese drei Technologien eine wirklich

AWS Certified Developer © Stephane Maarek

bbb sunny

AWS CloudWatch Metrics



- CloudWatch provides metrics for every services in AWS
- **Metric** is a variable to monitor (CPUUtilization, NetworkIn...)
- Metrics belong to **namespaces**
- **Dimension** is an attribute of a metric (instance id, environment, etc...).
- Up to 10 dimensions per metric
- Metrics have **timestamps**
- Can create CloudWatch dashboards of metrics

einen visuellen Aspekt der Funktionsweise unserer Umgebung erhalten.

AWS Certified Developer © Stephane Maarek

bbb sunny

AWS CloudWatch EC2 Detailed monitoring

- EC2 instance metrics have metrics “every 5 minutes”
- With detailed monitoring (for a cost), you get data “every 1 minute”
- Use detailed monitoring if you want to more prompt scale your ASG!

- The AWS Free Tier allows us to have 10 detailed monitoring metrics

- Note: EC2 Memory usage is by default not pushed (must be pushed from inside the instance as a custom metric)

müssen Sie sie als benutzerdefinierte Metrik aus der Instanz herauspushen.



AWS CloudWatch Custom Metrics

- Possibility to define and send your own custom metrics to CloudWatch
- Ability to use dimensions (attributes) to segment metrics
 - Instance.id
 - Environment.name
- Metric resolution:
 - Standard: 1 minute
 - High Resolution: up to 1 second (`StorageResolution` API parameter) – Higher cost
- Use API call `PutMetricData`
- Use exponential back off in case of throttle errors

Wenn Sie also zu viele Metrics an CloudWatch

AWS CloudWatch Alarms



- Alarms are used to trigger notifications for any metric
- Alarms can go to Auto Scaling, EC2 Actions, SNS notifications
- Various options (sampling, %, max, min, etc...)
- Alarm States:
 - OK
 - INSUFFICIENT_DATA
 - ALARM
- Period:
 - Length of time in seconds to evaluate the metric
 - High resolution custom metrics: can only choose 10 sec or 30 sec

Lassen Sie uns kurz in CloudWatch einsteigen und sehen, wie sie funktionieren.

AWS Certified Developer © Stephane Maarek

Überprüfen

AWS CloudWatch Logs

- Applications can send logs to CloudWatch using the SDK
- CloudWatch can collect log from:
 - Elastic Beanstalk: collection of logs from application
 - ECS: collection from containers
 - AWS Lambda: collection from function logs
 - VPC Flow Logs: VPC specific logs
 - API Gateway
 - CloudTrail based on filter
 - CloudWatch log agents: for example on EC2 machines
 - Route53: Log DNS queries
- CloudWatch Logs can go to:
 - Batch exporter to S3 for archival
 - Stream to ElasticSearch cluster for further analytics

verarbeitet werden, wenn Sie dies wünschen.

AWS Certified Developer © Stephane Maarek

Überprüfen

200. AWS CloudWatch Logs

AWS CloudWatch Logs

- CloudWatch Logs can use filter expressions
- Logs storage architecture:
 - Log groups: arbitrary name, usually representing an application
 - Log stream: instances within application / log files / containers
- Can define log expiration policies (never expire, 30 days, etc..)
- Using the AWS CLI we can tail CloudWatch logs
- To send logs to CloudWatch, make sure IAM permissions are correct!
- Security: encryption of logs using KMS at the Group Level

Protokollgruppe und sagen, dass die Protokollgruppe mit KMS verschlüsselt ist.

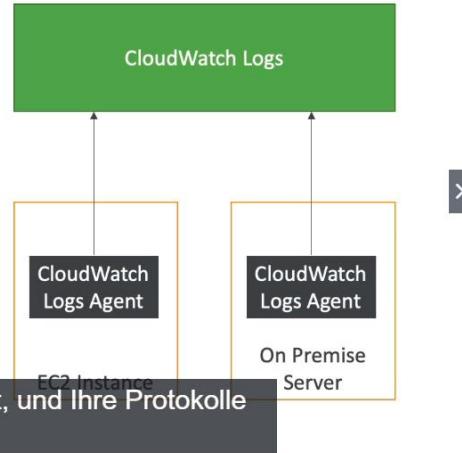


201. CloudWatch Agent & CloudWatch Logs Agent

CloudWatch Logs for EC2

- By default, no logs from your EC2 machine will go to CloudWatch
- You need to run a CloudWatch agent on EC2 to push the log files you want
- Make sure IAM permissions are correct
- The CloudWatch log agent can be setup on-premises too

sich um ein kleines Linux-Programm handelt, und Ihre Protokolle werden auch in cleveren Slugs enden.



CloudWatch Logs Agent & Unified Agent

- For virtual servers (EC2 instances, on-premise servers...)
- CloudWatch Logs Agent
 - Old version of the agent
 - Can only send to CloudWatch Logs
- CloudWatch Unified Agent
 - Collect additional system-level metrics such as RAM, processes, etc...
 - Collect logs to send to CloudWatch Logs
 - Centralized configuration using SSM Parameter Store

die vorherigen Agenten nicht hatten.

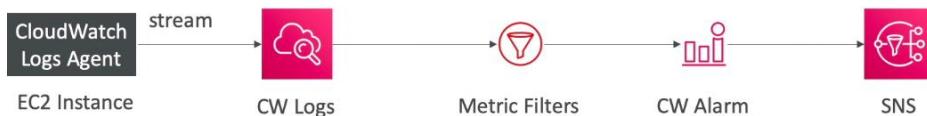
CloudWatch Unified Agent – Metrics

- Collected directly on your Linux server / EC2 instance
- CPU (active, guest, idle, system, user, steal)
- Disk metrics (free, used, total), Disk IO (writes, reads, bytes, iops)
- RAM (free, inactive, used, total, cached)
- Netstat (number of TCP and UDP connections, net packets, bytes)
- Processes (total, dead, bloqued, idle, running, sleep)
- Swap Space (free, used, used %)
- Reminder: out-of-the box metrics for EC2 – disk, CPU, network (high level)

OK, das war's für mich.

CloudWatch Logs Metric Filter

- CloudWatch Logs can use filter expressions
 - For example, find a specific IP inside of a log
 - Or count occurrences of “ERROR” in your logs
 - Metric filters can be used to trigger alarms
- Filters do not retroactively filter data. Filters only publish the metric data points for events that happen after the filter was created.



Gehen wir also zum Anfassen, um zu sehen, wie das funktioniert.

AWS CloudWatch Events

- Schedule: Cron jobs
- Event Pattern: Event rules to react to a service doing something
 - Ex: CodePipeline state changes!
- Triggers to Lambda functions, SQS/SNS/Kinesis Messages
- CloudWatch Event creates a small JSON document to give information about the change

Okay, das wars für CloudWatch Event.

Screenshot of the AWS CloudWatch Management Console showing the 'Step 1: Create rule' page. The left sidebar shows various AWS services like CloudWatch, Alarms, Rules, and Metrics. The main area is titled 'Step 1: Create rule' with the sub-instruction 'Create rules to invoke Targets based on Events happening in your AWS environment.' Under 'Event Source', the 'Schedule' option is selected, showing a dropdown for 'Fixed rate of' (set to 5) and 'Hours' (dropdown menu open), and a field for 'Cron expression' (set to '0/5 * * * *'). Below this is a 'Show sample event(s)' button, which displays a JSON sample event. On the right, under 'Targets', a dropdown menu lists several target types: Lambda function, EC2 TerminateInstances API call, ECS task, Event bus in another AWS account, Firehose delivery stream, Kinesis stream, Lambda function, SNS topic, and SQS queue. At the bottom, there are 'Cancel' and 'Configure details' buttons.

Amazon EventBridge



- EventBridge is the next evolution of CloudWatch Events
- Default event bus: generated by AWS services (CloudWatch Events)
- Partner event bus: receive events from SaaS service or applications (Zendesk, DataDog, Segment, Auth0...)
- Custom Event buses: for your own applications
- Event buses can be accessed by other AWS accounts
- Rules: how to process the events (similar to CloudWatch Events)

Sie erstellen Regeln für: Wie möchten Sie diese Ereignisse verarbeiten?

© Stephane Maarek

blob storage

204. EventBridge Overview

Amazon EventBridge Schema Registry

- EventBridge can analyze the events in your bus and infer the schema
 - The Schema Registry allows you to generate code for your application, that will know in advance how data is structured in the event bus
 - Schema can be versioned

Schema details			
Schema name	Last modified	Schema ARN	Schema type
aws.codipeline@CodePipelineActionExecutionStateChange	Dec 1, 2019, 12:11 AM GMT	-	
		Schema registry	Number of versions
		aws.events	1
Description			
Schema for event type CodePipelineActionExecutionStateChange, published by AWS service aws.codipeline			
Version 1 Created on Dec 1, 2019, 12:11 AM GMT			
Action	Download code bindings		
1 {			
2 "events": [
3 "data": {			
4 "version": "1.0",			
5 "title": "CodePipelineActionExecutionEventVersionAndStatusChange"			
6 },			
7 "path": "/",			
8 "opoulos": {			
9 "reblocks": {			
10 "ARN": "event", {			

Amazon EventBridge vs CloudWatch Events

- Amazon EventBridge builds upon and extends CloudWatch Events.
 - It uses the same service API and endpoint, and the same underlying service infrastructure.
 - EventBridge allows extension to add event buses for your custom applications and your third-party SaaS apps.
 - Event Bridge has the Schema Registry capability
 - EventBridge has a different name to mark the new capabilities
 - Over time, the CloudWatch Events name will be replaced with EventBridge

AWS X-Ray



- Debugging in Production, the good old way:
 - Test locally
 - Add log statements everywhere
 - Re-deploy in production
- Log formats differ across applications using CloudWatch and analytics is hard.
- Debugging: monolith “easy”, distributed services “hard”
- No common views of your entire architecture!
- Enter... AWS X-Ray!

AWS Certified Developer © Stephane Maarek

Hier kommt also AWS X-Ray.

by Stephane Maarek

AWS X-Ray Visual analysis of our applications



Natürlich können Sie noch mehr tun, aber Sie bekommen die Idee.

AWS Certified Developer © Stephane Maarek

by Stephane Maarek

AWS X-Ray advantages

- Troubleshooting performance (bottlenecks)
- Understand dependencies in a microservice architecture
- Pinpoint service issues
- Review request behavior
- Find errors and exceptions
- Are we meeting time SLA?
- Where I am throttled?
- Identify users that are impacted

Schließlich können wir wissen, welche Benutzer von unseren Fehlern betroffen sind,

AWS Certified Developer © Stephane Maarek

bbb sunny

X-Ray compatibility

- AWS Lambda
- Elastic Beanstalk
- ECS
- ELB
- API Gateway
- EC2 Instances or any application server (even on premise)

sein und für jede Anwendung geeignet zu sein.

AWS Certified Developer © Stephane Maarek

bbb sunny

AWS X-Ray Leverages Tracing

- Tracing is an end to end way to following a “request”
- Each component dealing with the request adds its own “trace”
- Tracing is made of segments (+ sub segments)
- Annotations can be added to traces to provide extra-information
- Ability to trace:
 - Every request
 - Sample request (as a % for example or a rate per minute)
- X-Ray Security:
 - IAM for authorization
 - KMS for encryption at rest

AWS Certified Developer © Stephane Maarek

können KMS zur Verschlüsselung im Ruhezustand verwenden.

by sunny

AWS X-Ray How to enable it?

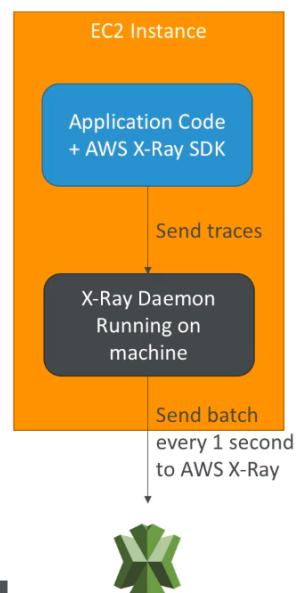
1) Your code (Java, Python, Go, Node.js, .NET) must import the AWS X-Ray SDK

- Very little code modification needed
 - The application SDK will then capture:
 - Calls to AWS services
 - HTTP / HTTPS requests
 - Database Calls (MySQL, PostgreSQL, DynamoDB)
 - Queue calls (SQS)
- 2) Install the X-Ray daemon or enable X-Ray AWS Integration
- X-Ray daemon works as a low level UDP packet interceptor (Linux / Windows / Mac...)
 - AWS Lambda / other AWS services already run the X-Ray daemon for you
 - Each application must have the IAM rights to write data to X-Ray

AWS Certified Developer © Stephane Maarek

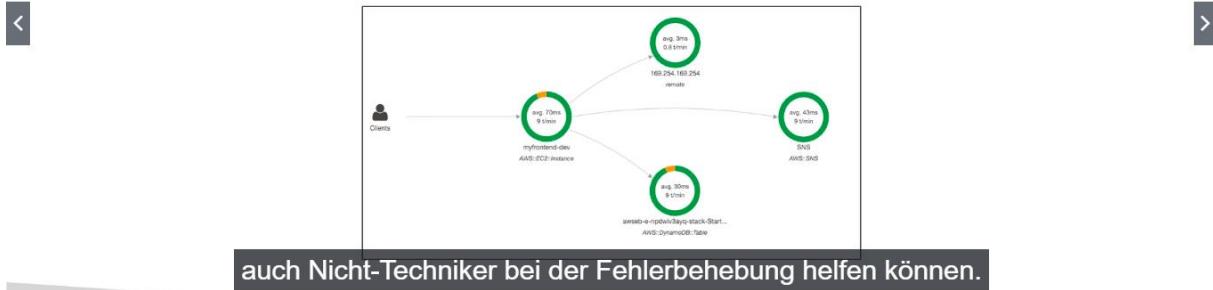
jede Sekunde einen Stapel an AWS X-Ray.

by sunny



The X-Ray magic

- X-Ray service collects data from all the different services
- Service map is computed from all the segments and traces
- X-Ray is graphical, so even non technical people can help troubleshoot



AWS X-Ray Troubleshooting

- If X-Ray is not working on EC2
 - Ensure the EC2 IAM Role has the proper permissions
 - Ensure the EC2 instance is running the X-Ray Daemon
- To enable on AWS Lambda:
 - Ensure it has an IAM execution role with proper policy (AWSX-RayWriteOnlyAccess)
 - Ensure that X-Ray is imported in the code

Kleiner Vorgesmack

X-Ray Instrumentation in your code

- Instrumentation means the measure of product's performance, diagnose errors, and to write trace information.
- To instrument your application code, you use the X-Ray SDK
- Many SDK require only configuration changes
- You can modify your application code to customize and annotation the data that the SDK sends to X-Ray, using interceptors, filters, handlers, middleware...

Example for Node.js & Express

```
var app = express();

var AWSXRay = require('aws-xray-sdk');
app.use(AWSXRay.express.openSegment('MyApp'));

app.get('/', function (req, res) {
| res.render('index');
});

app.use(AWSXRay.express.closeSegment());
```

Und dafür können wir Interceptors, Filter,



X-Ray Concepts

- Segments: each application / service will send them
- Subsegments: if you need more details in your segment
- Trace: segments collected together to form an end-to-end trace
- Sampling: decrease the amount of requests sent to X-Ray, reduce cost
- Annotations: Key Value pairs used to index traces and use with filters
- Metadata: Key Value pairs, not indexed, not used for searching
- The X-Ray daemon / agent has a config to send traces cross account:
 - make sure the IAM permissions are correct – the agent will assume the role
 - This allows to have a central account for all your application tracing

ein zentrales Konto für die gesamte Protokollierung und
Anwendungsverfolgung.

X-Ray Sampling Rules

- With sampling rules, you control the amount of data that you record
- You can modify sampling rules without changing your code
- By default, the X-Ray SDK records the first request **each second**, and **five percent** of any additional requests.
- **One request per second is the reservoir**, which ensures that at least one trace is recorded each second as long the service is serving requests.
- **Five percent is the rate** at which additional requests beyond the reservoir size are sampled.

Reservoirgröße hinaus abgetastet werden.

© Stephane Maarek

X-Ray Custom Sampling Rules

- You can create your own rules with the **reservoir** and **rate**

Example Higher minimum rate for POSTs

- Rule name – **POST minimum**
- Priority – **100**
- Reservoir – **10**
- Rate – **0.10**
- Service name – *
- Service type – *
- Host – *
- HTTP method – **POST**
- URL path – *
- Resource ARN – *

Example Debugging rule to trace all requests for a problematic route

A high-priority rule applied temporarily for debugging.

- Rule name – **DEBUG – history updates**
- Priority – **1**
- Reservoir – **1**
- Rate – **1**
- Service name – **Scorekeep**
- Service type – *
- Host – *
- HTTP method – **PUT**
- URL path – **/history/***
- Resource ARN – *

was los ist.

© Stephane Maarek

X-Ray Write APIs (used by the X-Ray daemon)

```
"Effect": "Allow",
"Action": [
    "xray:PutTraceSegments",
    "xray:PutTelemetryRecords",
    "xray:GetSamplingRules",
    "xray:GetSamplingTargets",
    "xray:GetSamplingStatisticSummaries"
],
"Resource": [
    "*"
]
```

arn:aws:iam::123456789012:policy/XRayWritePolicy
X-Ray-Daemon über die richtige IAM-Richtlinie verfügt, die diese API-Aufrufe autorisiert.

- PutTraceSegments: Uploads segment documents to AWS X-Ray
- PutTelemetryRecords: Used by the AWS X-Ray daemon to upload telemetry.
 - SegmentsReceivedCount, SegmentsRejectedCounts, BackendConnectionErrors...
- GetSamplingRules: Retrieve all sampling rules (to know what/when to send)
- GetSamplingTargets & GetSamplingStatisticSummaries: advanced
- The X-Ray daemon needs to have an IAM policy authorizing the correct API calls to



X-Ray Read APIs – continued

```
"Effect": "Allow",
"Action": [
    "xray:GetSamplingRules",
    "xray:GetSamplingTargets",
    "xray:GetSamplingStatisticSummaries",
    "xray:BatchGetTraces",
    "xray:GetServiceGraph",
    "xray:GetTraceGraph",
    "xray:GetTraceSummaries",
    "xray:GetGroups",
    "xray:GetGroup",
    "xray:GetTimeSeriesServiceStatistics"
],
"Resource": [
    "*"
]
```

- GetServiceGraph: main graph
- BatchGetTraces: Retrieves a list of traces specified by ID. Each trace is a collection of segment documents that originates from a single request.
- GetTraceSummaries: Retrieves IDs and annotations for traces available for a specified time frame using an optional filter. To get the full traces, pass the trace IDs to BatchGetTraces.
- GetTraceGraph: Retrieves a service graph for one or more specific trace IDs.



X-Ray with Elastic Beanstalk



- AWS Elastic Beanstalk platforms include the X-Ray daemon
- You can run the daemon by setting an option in the Elastic Beanstalk console or with a configuration file (in .ebextensions/xray-daemon.config)

```
option_settings:  
  aws:elasticbeanstalk:xray:  
    | XRayEnabled: true
```

- Make sure to give your instance profile the correct IAM permissions so that the X-Ray daemon can function correctly
- Then make sure your application code is instrumented with the X-Ray SDK
- Note: The X-Ray daemon is not provided for Multicontainer Docker

© Stephane Maarek

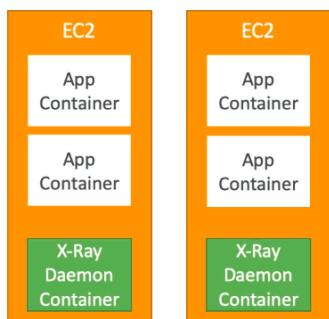
den X-Ray-Daemon selbst verwalten, wie wir

↓↓↓

ECS + X-Ray integration options

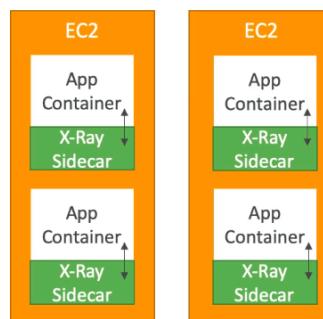
ECS Cluster

X-Ray Container as a Daemon



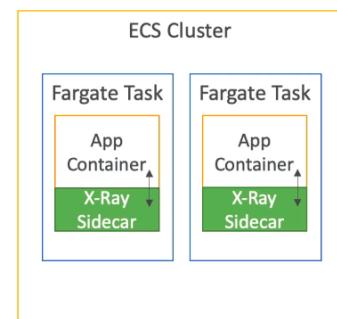
ECS Cluster

X-Ray Container as a "Side Car"



Fargate Cluster

X-Ray Container as a "Side Car"



Jeder hat eine Röntgenaufnahme und hoffentlich macht das ein bisschen klarer, wie man das machen könnte.

AWS Certified Developer © Stephane Maarek

↓↓↓

ECS + X-Ray: Example Task Definition

```
{  
    "name": "xray-daemon",  
    "image": "123456789012.dkr.ecr.us-east-2.amazonaws.com/xray-daemon",  
    "cpu": 32,  
    "memoryReservation": 256,  
    "portMappings": [  
        {  
            "hostPort": 0,  
            "containerPort": 2000,  
            "protocol": "udp"  
        }  
    ],  
    {  
        "name": "scorekeep-api",  
        "image": "123456789012.dkr.ecr.us-east-2.amazonaws.com/scorekeep-api",  
        "cpu": 192,  
        "memoryReservation": 512,  
        "environment": [  
            { "name": "AWS_REGION", "value": "us-east-2" },  
            { "name": "NOTIFICATION_TOPIC", "value": "arn:aws:sns:us-east-2:123456789012:scorekeep-notifications" },  
            { "name": "AWS_XRAY_DAEMON_ADDRESS", "value": "xray-daemon:2000" }  
        ],  
        "portMappings": [  
            {  
                "hostPort": 5000,  
                "containerPort": 5000  
            }  
        ]  
    },  
    "links": [  
        "xray-daemon"  
    ]  
}
```

<https://docs.aws.amazon.com/xray/latest/devguide/xray-daemon-ecs.html#xray-daemon-ecs-build>

AWS Certified Developer © Stephane Maarek

bbb.talks

Und schließlich müssen diese beiden Container vernetzt werden.

AWS CloudTrail



- Provides governance, compliance and audit for your AWS Account
- CloudTrail is enabled by default!
- Get an history of events / API calls made within your AWS Account by:
 - Console
 - SDK
 - CLI
 - AWS Services
- Can put logs from CloudTrail into CloudWatch Logs
- If a resource is deleted in AWS, look into CloudTrail first!

eine Vorstellung davon bekommen, wie es funktioniert.

AWS Certified Developer © Stephane Maarek

bbb.talks

CloudTrail vs CloudWatch vs X-Ray

- CloudTrail:

- Audit API calls made by users / services / AWS console
- Useful to detect unauthorized calls or root cause of changes

- CloudWatch:

- CloudWatch Metrics over time for monitoring
- CloudWatch Logs for storing application log
- CloudWatch Alarms to send notifications in case of unexpected metrics

- X-Ray:

- Automated Trace Analysis & Central Service Map Visualization
- Latency, Errors and Fault analysis
- Request tracking across distributed systems

Sie können, wie gesagt, auch eine Anforderungsverfolgung für
Ihre digitalen

AWS Certified Developer © Stephane Maarek

15 Seiten

217. Introduction to Messaging

Section Introduction

- When we start deploying multiple applications, they will inevitably need to communicate with one another
- There are two patterns of application communication



AWS Certified Developer © Stephane Maarek
135 / 235

Da sie

Speaker icon, CC icon, gear icon, etc.

Section Introduction

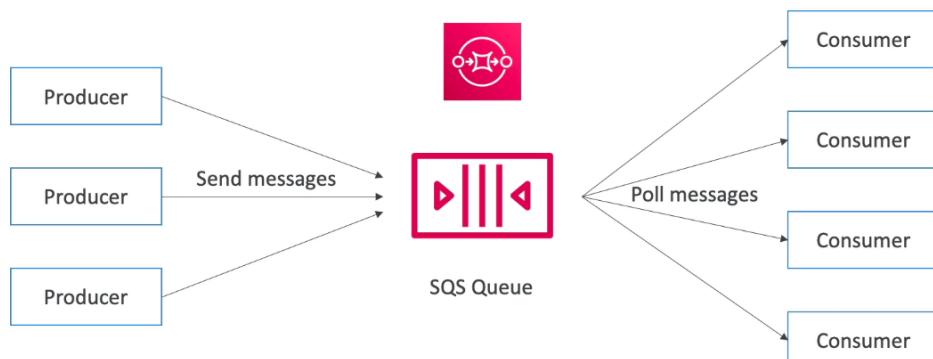
- Synchronous between applications can be problematic if there are sudden spikes of traffic
- What if you need to suddenly encode 1000 videos but usually it's 10?
- In that case, it's better to **decouple** your applications,
 - using SQS: queue model
 - using SNS: pub/sub model
 - using Kinesis: real-time streaming model
- These services can scale independently from our application!

AWS Certified Developer © Stephane Maarek

werden, ist, dass unsere Services jetzt

BB Sammy

Amazon SQS What's a queue?



© Stephane Maarek

Möglicherweise verbrauchen mehrere Verbraucher Nachrichten aus einer Sternchenwarteschlange.

BB Sammy

Amazon SQS – Standard Queue



- Oldest offering (over 10 years old)
- Fully managed service, used to decouple applications
- Attributes:
 - Unlimited throughput, unlimited number of messages in queue
 - Default retention of messages: 4 days, maximum of 14 days
 - Low latency (<10 ms on publish and receive)
 - Limitation of 256KB per message sent
- Can have duplicate messages (at least once delivery, occasionally)
- Can have out of order messages (best effort ordering)

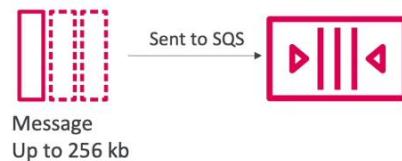
Es kann auch Nachrichten außerhalb der Bestellung enthalten,
was bedeutet, dass die Bestellung am besten ist.

© Stephane Maarek

by Sammy

SQS – Producing Messages

- Produced to SQS using the SDK (SendMessage API)
- The message is persisted in SQS until a consumer deletes it
- Message retention: default 4 days, up to 14 days
- Example: send an order to be processed
 - Order id
 - Customer id
 - Any attributes you want
- SQS standard: unlimited throughput



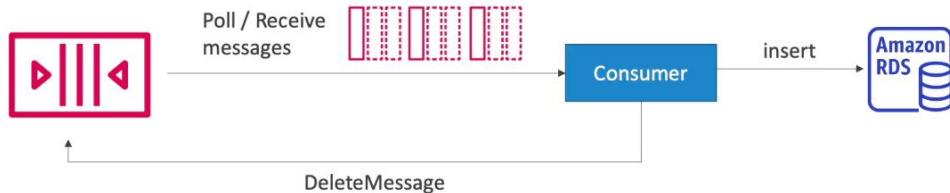
© Stephane Maarek

Wir haben also etwas über Produzenten gesehen.

by Sammy

SQS – Consuming Messages

- Consumers (running on EC2 instances, servers, or AWS Lambda)...
- Poll SQS for messages (receive up to 10 messages at a time)
- Process the messages (example: insert the message into an RDS database)
- Delete the messages using the DeleteMessage API

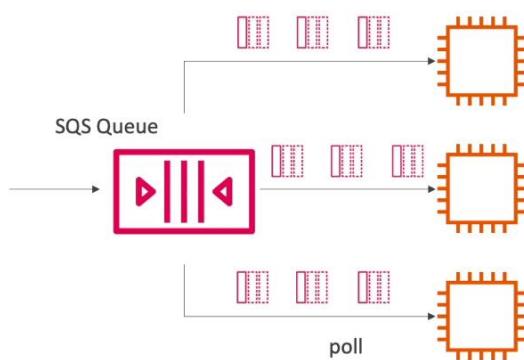


© Stephane Maarek

Und deshalb ist die Nachrichtenverarbeitung vollständig.

by Sammy

SQS – Multiple EC2 Instances Consumers

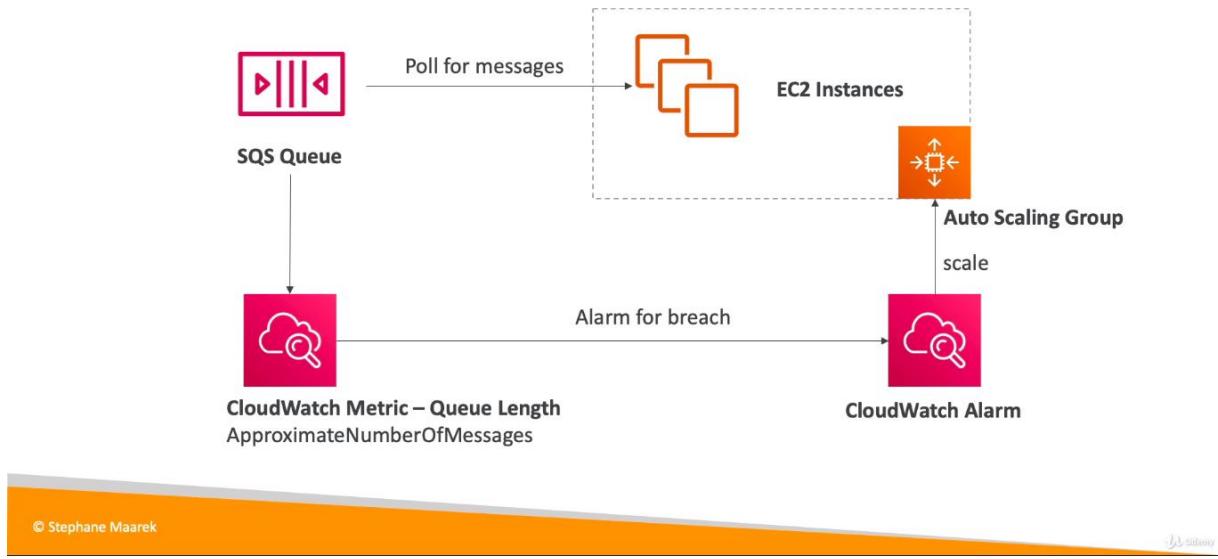


- Consumers receive and process messages in parallel
- At least once delivery
- Best-effort message ordering
- Consumers delete messages after processing them
- We can scale consumers horizontally to improve throughput of processing

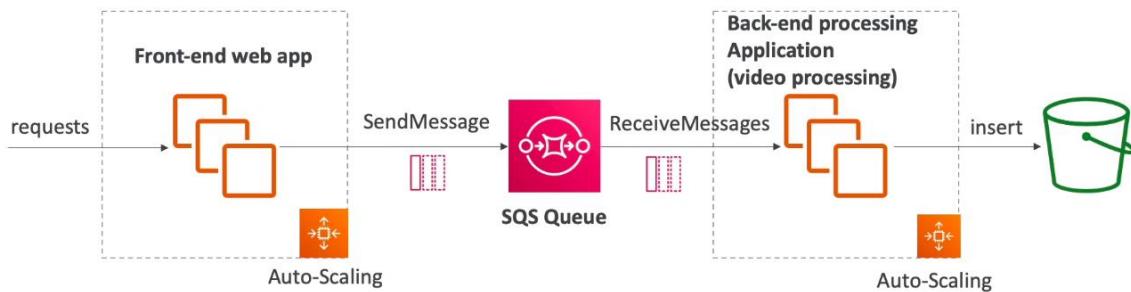
© Stephane Maarek

by Sammy

SQS with Auto Scaling Group (ASG)

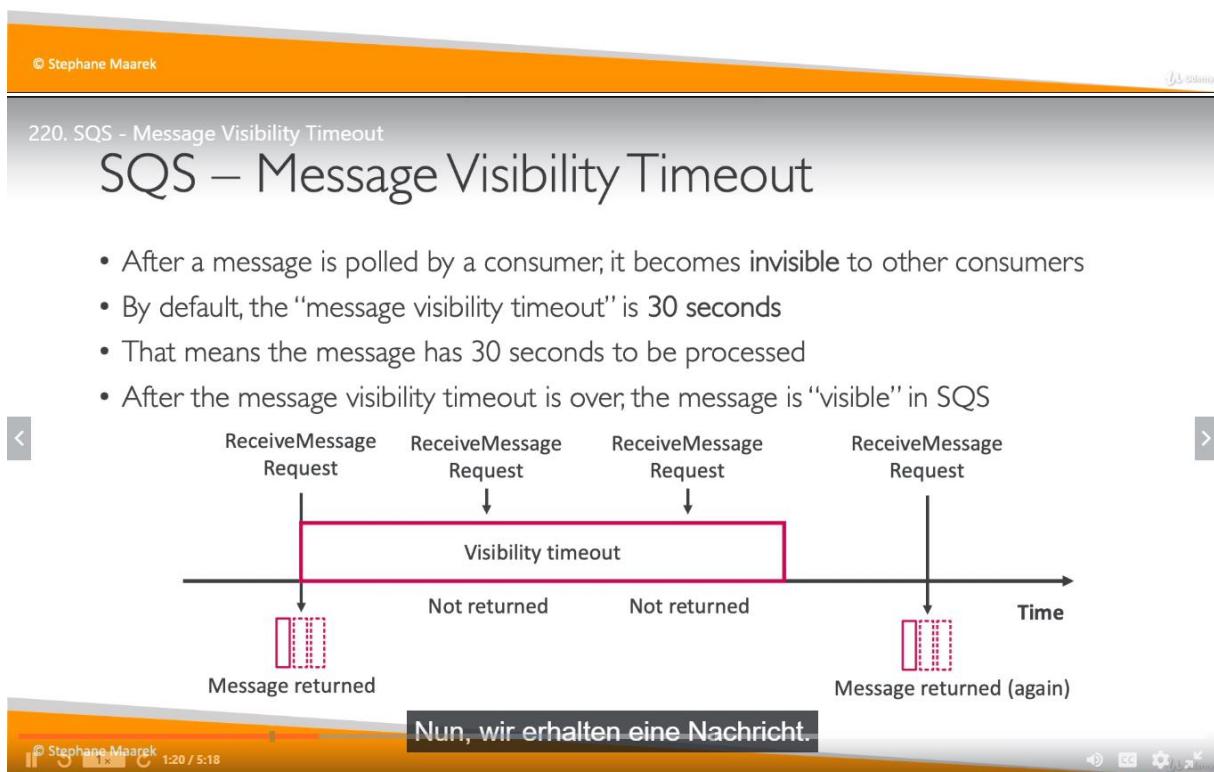


SQS to decouple between application tiers



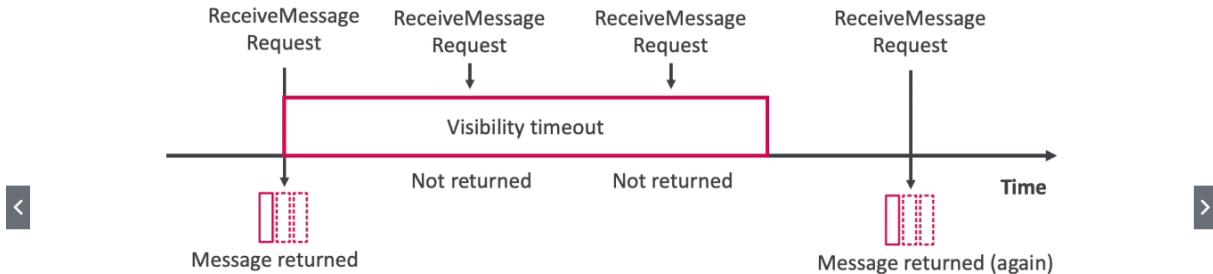
Amazon SQS - Security

- **Encryption:**
 - In-flight encryption using HTTPS API
 - At-rest encryption using KMS keys
 - Client-side encryption if the client wants to perform encryption/decryption itself
- **Access Controls:** IAM policies to regulate access to the SQS API
- **SQS Access Policies** (similar to S3 bucket policies)
 - Useful for cross-account access to SQS queues
 - Useful for allowing other services (SNS, S3...) to write to an SQS queue



220. SQS - Message Visibility Timeout

SQS – Message Visibility Timeout

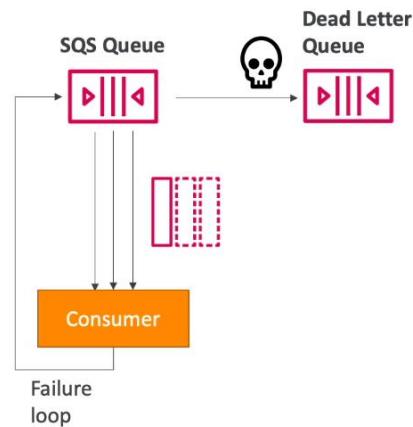


- If a message is not processed within the visibility timeout, it will be processed twice
- A consumer could call the `ChangeMessageVisibility` API to get more time
- If visibility timeout is high (hours), and consumer crashes, re-processing will take time
- If visibility timeout is too low (seconds) may get duplicates

221. SQS - Dead Letter Queues

Amazon SQS – Dead Letter Queue

- If a consumer fails to process a message within the Visibility Timeout...
the message goes back to the queue!
- We can set a threshold of how many times a message can go back to the queue
- After the `MaximumReceives` threshold is exceeded, the message goes into a dead letter queue (DLQ)
- Useful for debugging!
- Make sure to process the messages in the DLQ before they expire:
 - Good to set a retention of 14 days in the DLQ



Untertitel

Amazon SQS – Delay Queue

- Delay a message (consumers don't see it immediately) up to 15 minutes
- Default is 0 seconds (message is available right away)
- Can set a default at queue level
- Can override the default on send using the DelaySeconds parameter



© Stephane Maarek

223. SQS - Certified Developer concepts

Amazon SQS - Long Polling

When a consumer requests messages from the queue, it can optionally "wait" for messages to arrive if there are none in the queue

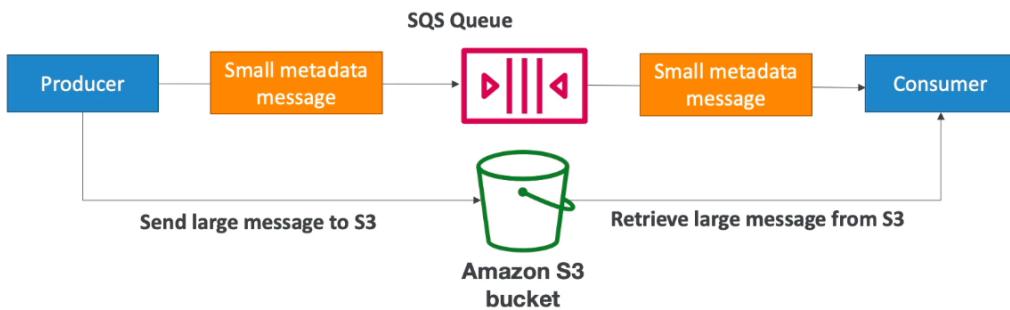
- This is called Long Polling
- LongPolling decreases the number of API calls made to SQS while increasing the efficiency and latency of your application.
- The wait time can be between 1 sec to 20 sec (20 sec preferable)
- Long Polling is preferable to Short Polling
- Long polling can be enabled at the queue level or at the API level using WaitTimeSeconds

```
graph TD; Message[message] --> Queue[SQS Queue]; Queue -- poll --> Consumer[Consumer]
```

The diagram illustrates the flow of data through an Amazon SQS Long Polling process. It starts with a 'message' box on the left, which has an arrow pointing down to a central 'SQS Queue' box. The 'SQS Queue' box contains a red icon with three vertical bars and a circular arrow. Below the 'SQS Queue' box is a circular timer icon with the word 'poll'. From the 'SQS Queue' box, an arrow labeled 'poll' points down to a 'Consumer' box on the right.

SQS Extended Client

- Message size limit is 256KB, how to send large messages, e.g. 1GB?
- Using the SQS Extended Client (Java Library)



© Stephane Maarek

JB Salomé

SQS – Must know API

- CreateQueue (MessageRetentionPeriod), DeleteQueue
 - PurgeQueue: delete all the messages in queue
 - SendMessage (DelaySeconds), ReceiveMessage, DeleteMessage
 - ReceiveMessageWaitTimeSeconds: Long Polling
 - ChangeMessageVisibility: change the message timeout
-
- Batch APIs for SendMessage, DeleteMessage, ChangeMessageVisibility helps decrease your costs

© Stephane Maarek

JB Salomé

Amazon SQS – FIFO Queue

- FIFO = First In First Out (ordering of messages in the queue)



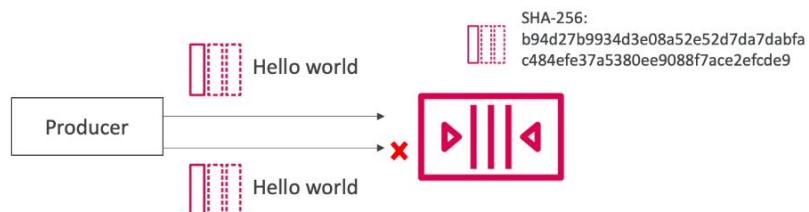
- Limited throughput: 300 msg/s without batching, 3000 msg/s with
- Exactly-once send capability (by removing duplicates)
- Messages are processed in order by the consumer

© Stephane Maarek

by Stéphane Maarek

SQS FIFO – Deduplication

- De-duplication interval is 5 minutes
- Two de-duplication methods:
 - Content-based deduplication: will do a SHA-256 hash of the message body
 - Explicitly provide a Message Deduplication ID



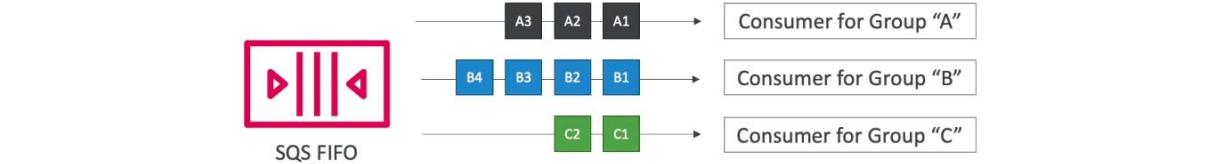
© Stephane Maarek

by Stéphane Maarek

225. SQS - FIFO Queues Advanced

SQS FIFO – Message Grouping

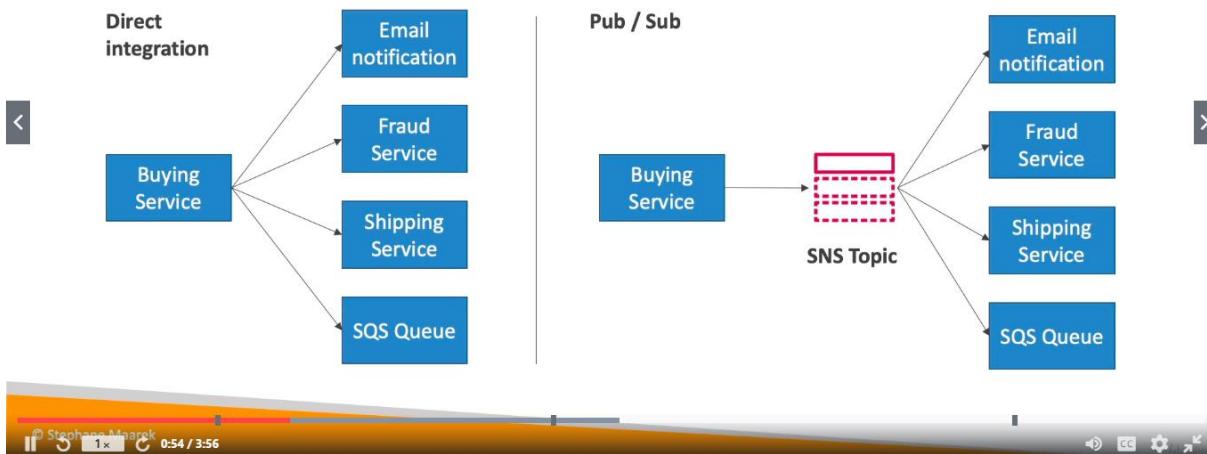
- If you specify the same value of **MessageGroupId** in an SQS FIFO queue, you can only have one consumer, and all the messages are in order
- To get ordering at the level of a subset of messages, specify different values for **MessageGroupId**
 - Messages that share a common Message Group ID will be in order within the group
 - Each Group ID can have a different consumer (parallel processing!)
 - Ordering across groups is not guaranteed



226. Amazon SNS - Overview

Amazon SNS

- What if you want to send one message to many receivers?



SNS integrates with a lot of AWS services

- Many AWS services can send data directly to SNS for notifications
- CloudWatch (for alarms)
- Auto Scaling Groups notifications
- Amazon S3 (on bucket events)
- CloudFormation (upon state changes => failed to build, etc)
- Etc...

© Stephane Maarek

JB Savary

AWS SNS – How to publish

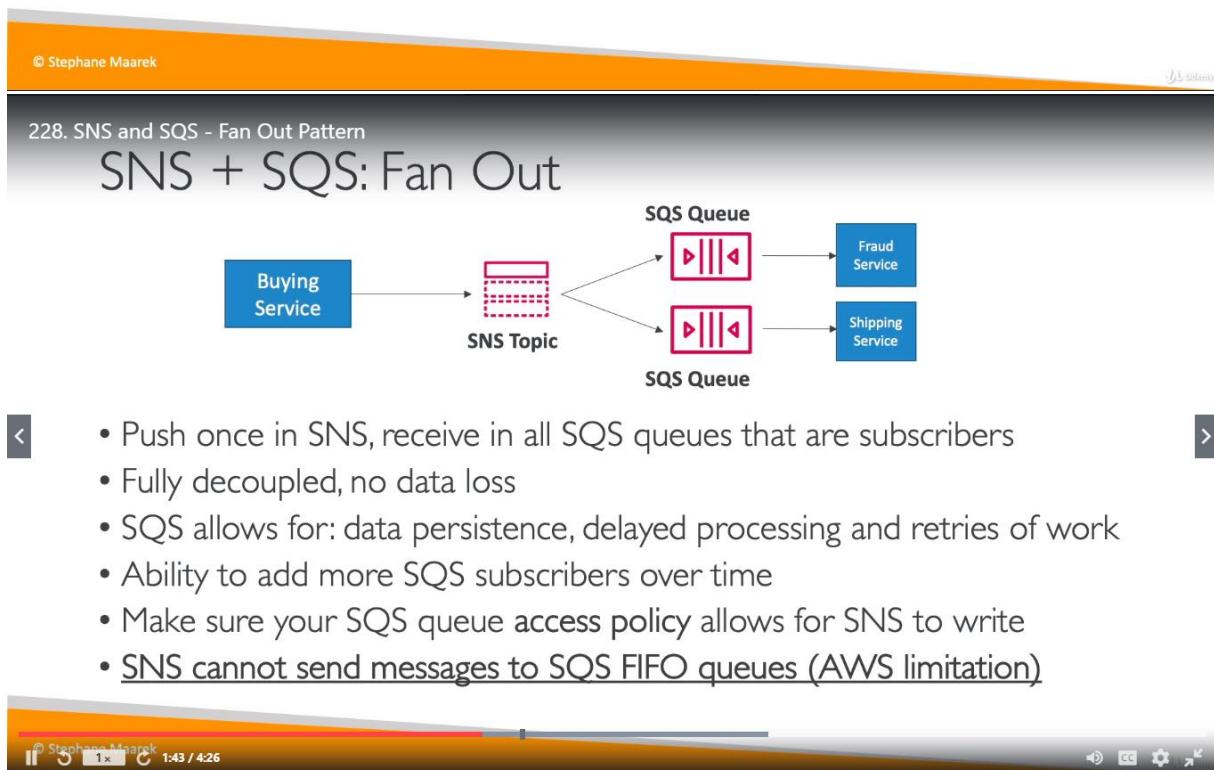
- Topic Publish (using the SDK)
 - Create a topic
 - Create a subscription (or many)
 - Publish to the topic
- Direct Publish (for mobile apps SDK)
 - Create a platform application
 - Create a platform endpoint
 - Publish to the platform endpoint
 - Works with Google GCM, Apple APNS, Amazon ADM...

© Stephane Maarek

JB Savary

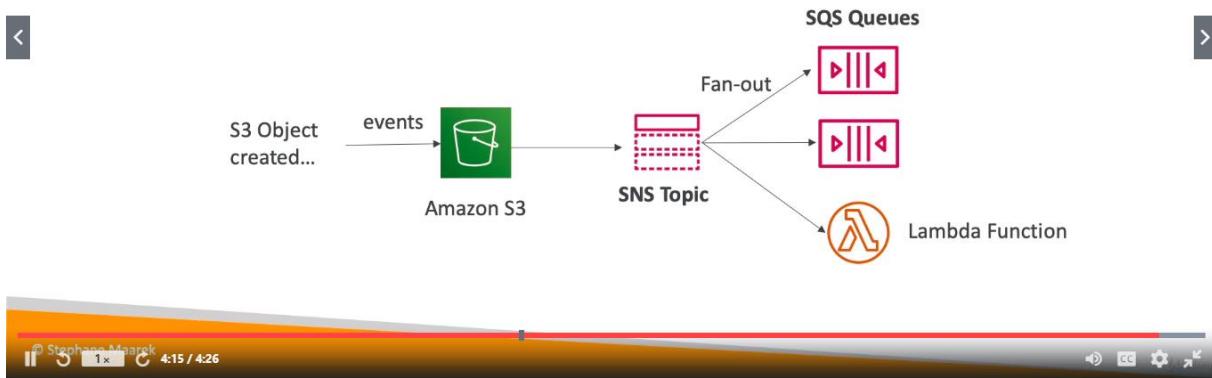
Amazon SNS – Security

- Encryption:
 - In-flight encryption using HTTPS API
 - At-rest encryption using KMS keys
 - Client-side encryption if the client wants to perform encryption/decryption itself
- Access Controls: IAM policies to regulate access to the SNS API
- SNS Access Policies (similar to S3 bucket policies)
 - Useful for cross-account access to SNS topics
 - Useful for allowing other services (S3...) to write to an SNS topic



Application: S3 Events to multiple queues

- For the same combination of: **event type** (e.g. object create) and **prefix** (e.g. images/) you can only have one S3 Event rule
- If you want to send the same S3 event to many SQS queues, use fan-out



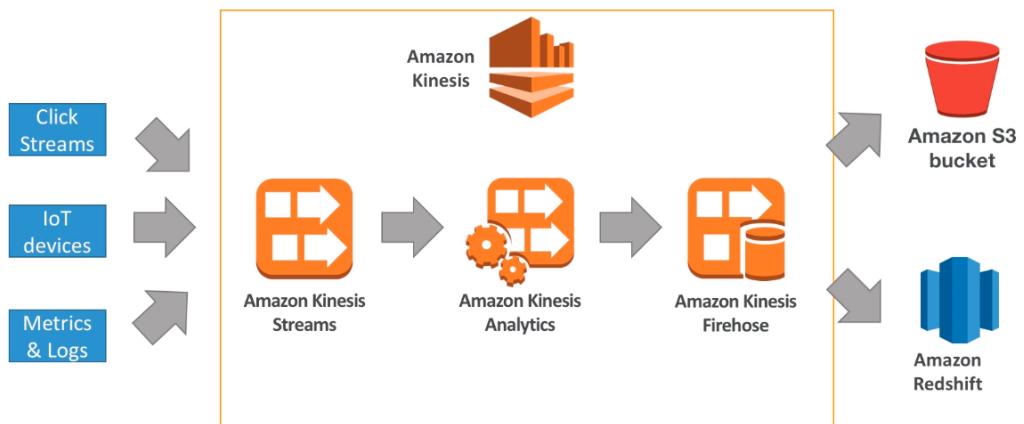
AWS Kinesis Overview



- **Kinesis** is a managed alternative to Apache Kafka
- Great for application logs, metrics, IoT, clickstreams
- Great for “real-time” big data
- Great for streaming processing frameworks (Spark, NiFi, etc...)
- Data is automatically replicated to 3 AZ

- **Kinesis Streams:** low latency streaming ingest at scale
- **Kinesis Analytics:** perform real-time analytics on streams using SQL
- **Kinesis Firehose:** load streams into S3, Redshift, ElasticSearch...

Kinesis



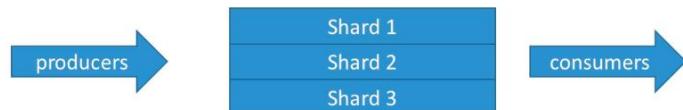
AWS Certified Developer © Stephane Maarek

3h 30m

229. AWS Kinesis Overview

Kinesis Streams Overview

- Streams are divided in ordered Shards / Partitions



- Data retention is 1 day by default, can go up to 7 days
- Ability to reprocess / replay data
- Multiple applications can consume the same stream
- Real-time processing with scale of throughput
- Once data is inserted in Kinesis, it can't be deleted (immutability)

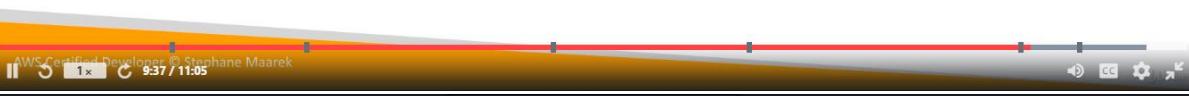
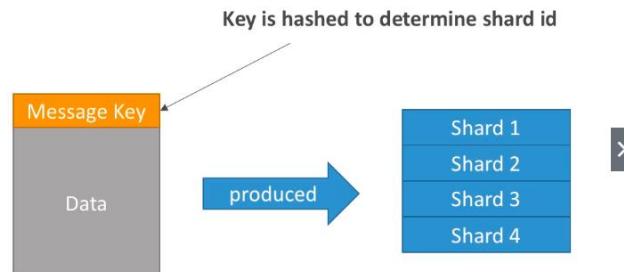
AWS Certified Developer © Stephane Maarek

3h 30m

229. AWS Kinesis Overview

AWS Kinesis API – Put records

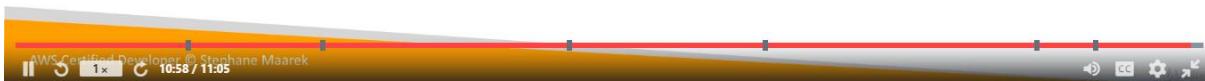
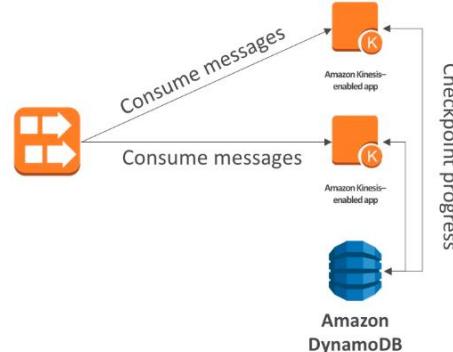
- PutRecord API + Partition key that gets hashed
- The same key goes to the same partition (helps with ordering for a specific key)
- Messages sent get a "sequence number"
- Choose a partition key that is highly distributed (helps prevent "hot partition")
 - user_id if many users
 - **Not country_id** if 90% of the users are in one country
- Use Batching with PutRecords to reduce costs and increase throughput
- ProvisionedThroughputExceeded if we go over the limits
- Can use CLI, AWS SDK, or producer libraries from various frameworks



229. AWS Kinesis Overview

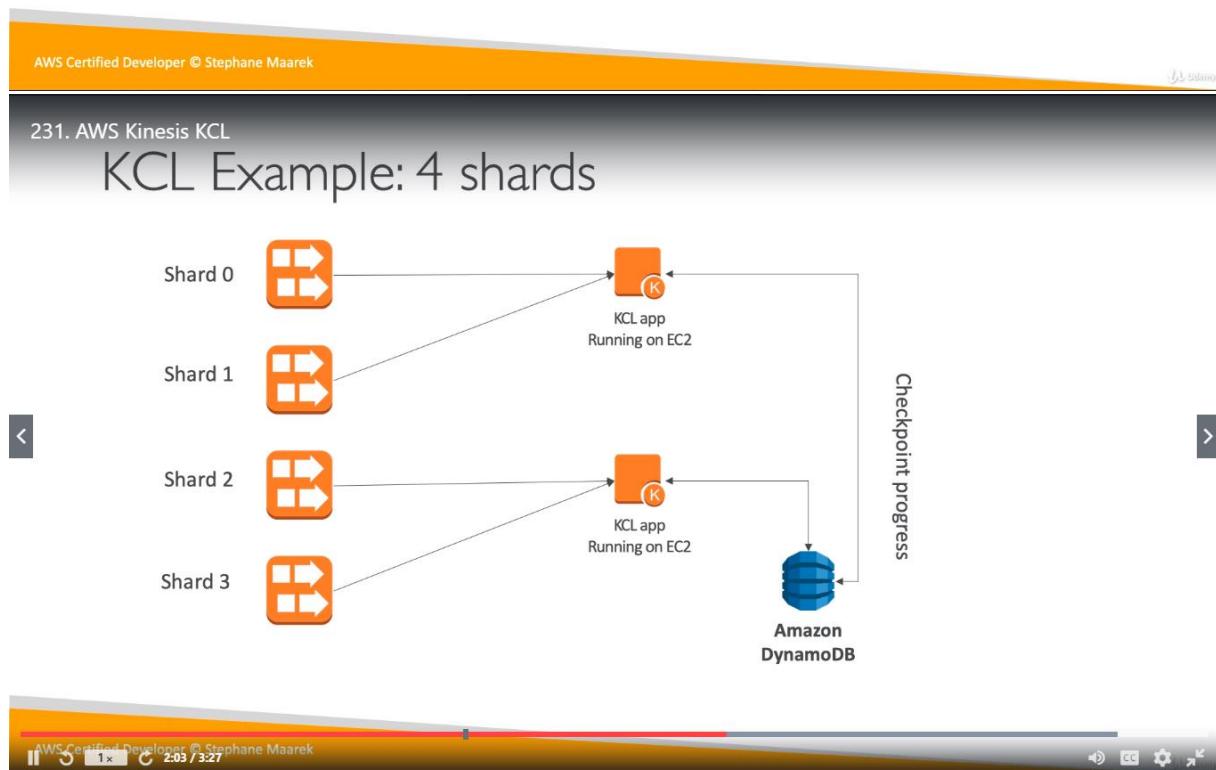
AWS Kinesis API – Consumers

- Can use a normal consumer (CLI, SDK, etc...)
- Can use Kinesis Client Library (in Java, Node, Python, Ruby, .Net)
 - KCL uses DynamoDB to checkpoint offsets
 - KCL uses DynamoDB to track other workers and share the work amongst shards



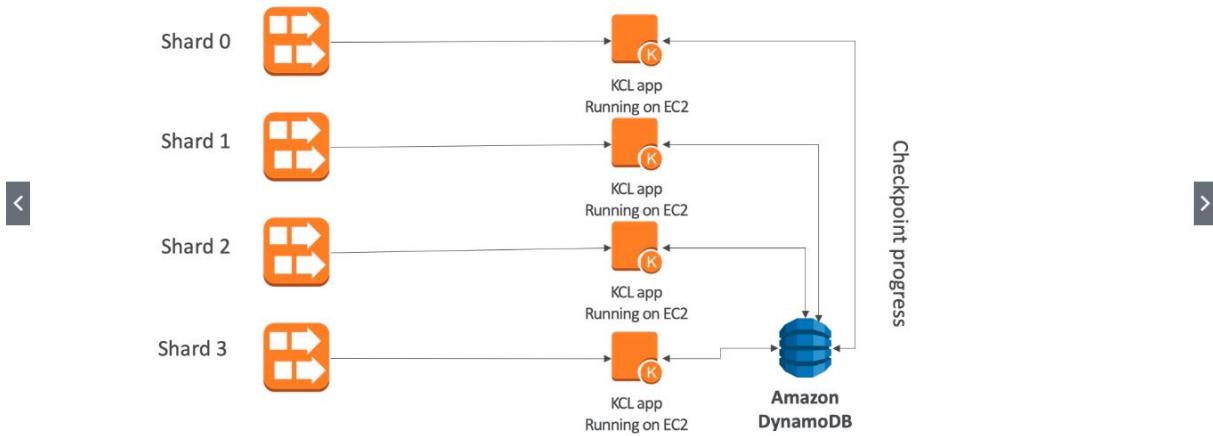
Kinesis KCL in Depth

- Kinesis Client Library (KCL) is Java library that helps read record from a Kinesis Streams with distributed applications sharing the read workload
- Rule: each shard is be read by only one KCL instance
- Means 4 shards = max 4 KCL instances
- Means 6 shards = max 6 KCL instances
- Progress is checkpointed into DynamoDB (need IAM access)
- KCL can run on EC2, Elastic Beanstalk, on Premise Application
- Records are read in order at the shard level



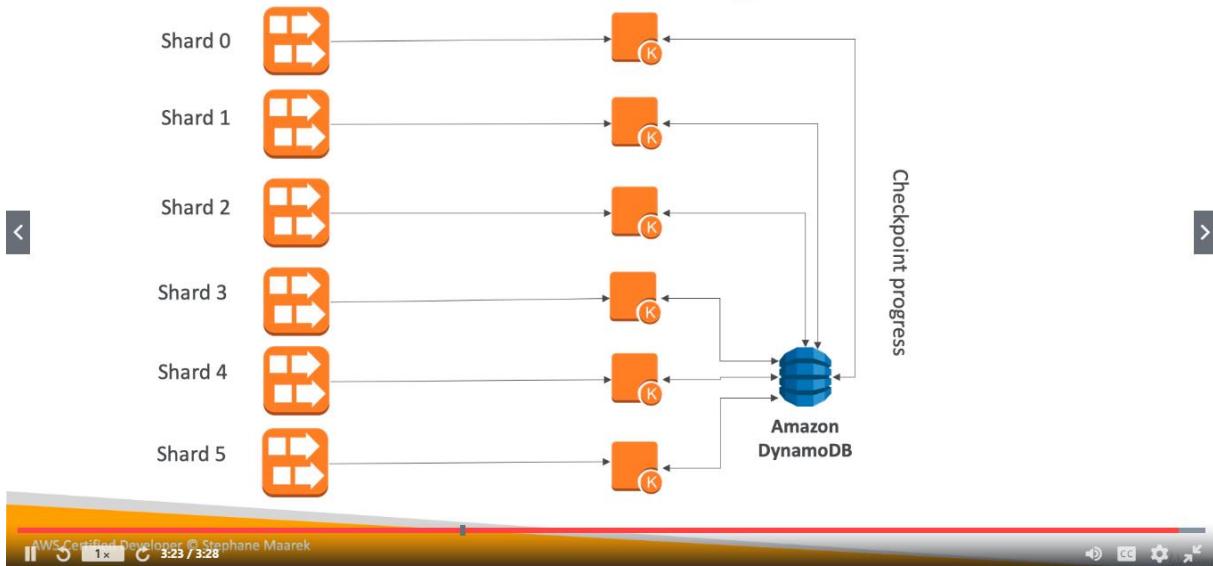
231. AWS Kinesis KCL

KCL Example: 4 shards, scaling KCL app



231. AWS Kinesis KCL

KCL Example: 6 shards, scaling KCL



Kinesis Security

- Control access / authorization using IAM policies
- Encryption in flight using HTTPS endpoints
- Encryption at rest using KMS
- Possibility to encrypt / decrypt data client side (harder)
- VPC Endpoints available for Kinesis to access within VPC

AWS Kinesis Data Analytics



- Perform real-time analytics on Kinesis Streams using SQL
- Kinesis Data Analytics:
 - Auto Scaling
 - Managed: no servers to provision
 - Continuous: real time
- Pay for actual consumption rate
- Can create streams out of the real-time queries

AWS Kinesis Firehose



- Fully Managed Service, no administration
- Near Real Time (60 seconds latency)
- Load data into Redshift / Amazon S3 / ElasticSearch / Splunk
- Automatic scaling
- Support many data formats (pay for conversion)
- Pay for the amount of data going through Firehose

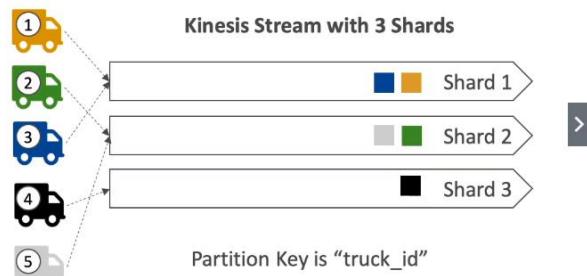
SQS vs SNS vs Kinesis

SQS:	SNS:	Kinesis:
<ul style="list-style-type: none">• Consumer “pull data”• Data is deleted after being consumed• Can have as many workers (consumers) as we want• No need to provision throughput• No ordering guarantee (except FIFO queues)• Individual message delay capability	<ul style="list-style-type: none">• Push data to many subscribers• Up to 10,000,000 subscribers• Data is not persisted (lost if not delivered)• Pub/Sub• Up to 100,000 topics• No need to provision throughput• Integrates with SQS for fan-out architecture pattern	<ul style="list-style-type: none">• Consumers “pull data”• As many consumers as we want• Possibility to replay data• Meant for real-time big data, analytics and ETL• Ordering at the shard level• Data expires after X days• Must provision throughput

234. Data Ordering for Kinesis vs SQS FIFO

Ordering data into Kinesis

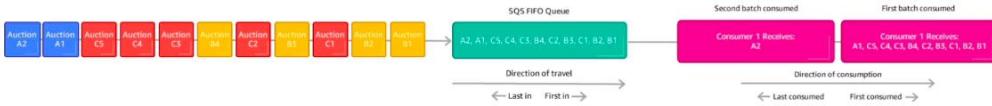
- Imagine you have 100 trucks (truck_1, truck_2, ... truck_100) on the road sending their GPS positions regularly into AWS.
- You want to consume the data in order for each truck, so that you can track their movement accurately.
- How should you send that data into Kinesis?
- Answer: send using a "Partition Key" value of the "truck_id"
- The same key will always go to the same shard



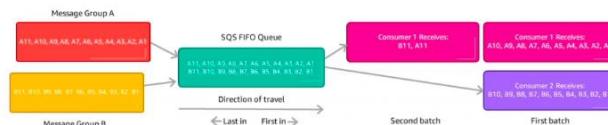
234. Data Ordering for Kinesis vs SQS FIFO

Ordering data into SQS

- For SQS standard, there is no ordering.
- For SQS FIFO, if you don't use a Group ID, messages are consumed in the order they are sent, with only one consumer



- You want to scale the number of consumers, but you want messages to be "grouped" when they are related to each other
- Then you use a Group ID (similar to Partition Key in Kinesis)



234. Data Ordering for Kinesis vs SQS FIFO

Kinesis vs SQS ordering

- Let's assume 100 trucks, 5 kinesis shards, 1 SQS FIFO
- Kinesis Data Streams:
 - On average you'll have 20 trucks per shard
 - Trucks will have their data ordered within each shard
 - The maximum amount of consumers in parallel we can have is 5
 - Can receive up to 5 MB/s of data
- SQS FIFO
 - You only have one SQS FIFO queue
 - You will have 100 Group ID
 - You can have up to 100 Consumers (due to the 100 Group ID)
 - You have up to 300 messages per second (or 3000 if using batching)

© Stephane Maarek

JB Savary

What's serverless?

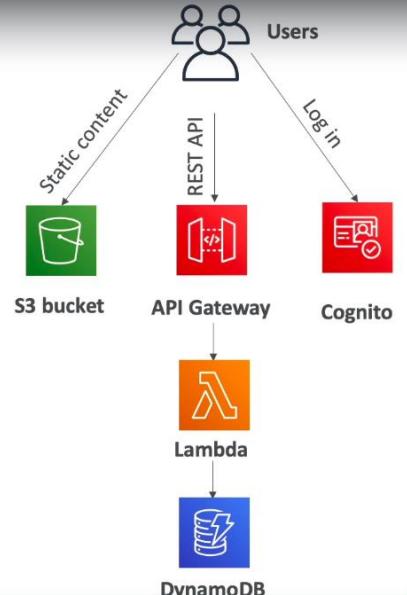
- Serverless is a new paradigm in which the developers don't have to manage servers anymore...
- They just deploy code
- They just deploy... functions !
- Initially... Serverless == FaaS (Function as a Service)
- Serverless was pioneered by AWS Lambda but now also includes anything that's managed: "databases, messaging, storage, etc."
- **Serverless does not mean there are no servers...**
it means you just don't manage / provision / see them

© Stephane Maarek

JB Savary

Serverless in AWS

- AWS Lambda
- DynamoDB
- AWS Cognito
- AWS API Gateway
- Amazon S3
- AWS SNS & SQS
- AWS Kinesis Data Firehose
- Aurora Serverless
- Step Functions
- Fargate



Why AWS Lambda



Amazon EC2

- Virtual Servers in the Cloud
- Limited by RAM and CPU
- Continuously running
- Scaling means intervention to add / remove servers



Amazon Lambda

- Virtual **functions** – no servers to manage!
- Limited by time - **short executions**
- Run **on-demand**
- **Scaling is automated!**



Benefits of AWS Lambda

- Easy Pricing:
 - Pay per request and compute time
 - Free tier of 1,000,000 AWS Lambda requests and 400,000 GBs of compute time
- Integrated with the whole AWS suite of services
- Integrated with many programming languages
- Easy monitoring through AWS CloudWatch
- Easy to get more resources per functions (up to 3GB of RAM!)
- Increasing RAM will also improve CPU and network!



AWS Lambda language support

- Node.js (JavaScript)
- Python
- Java (Java 8 compatible)
- C# (.NET Core)
- Golang
- C# / Powershell
- Ruby
- Custom Runtime API (community supported, example Rust)
- *Important: Docker is not for AWS Lambda, it's for ECS / Fargate*



AWS Lambda Integrations

Main ones



API Gateway



Kinesis



DynamoDB



S3



CloudFront



CloudWatch Events
EventBridge



CloudWatch Logs



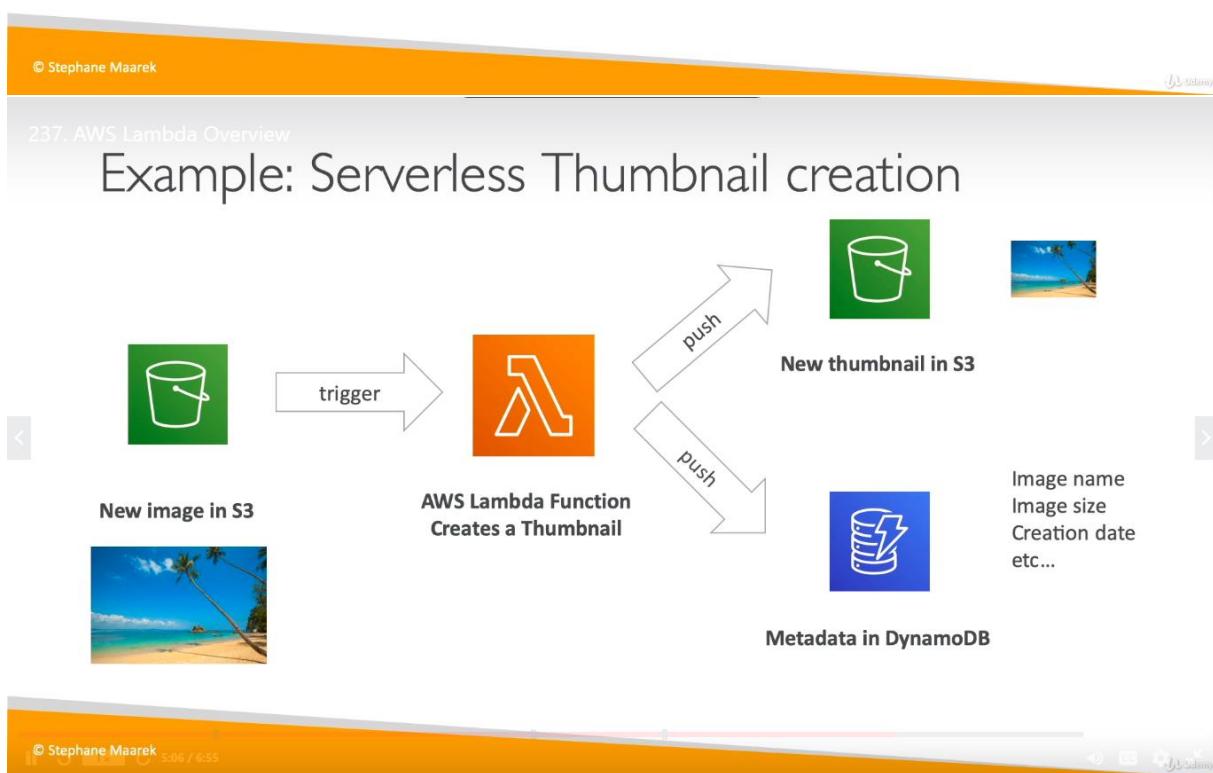
SNS



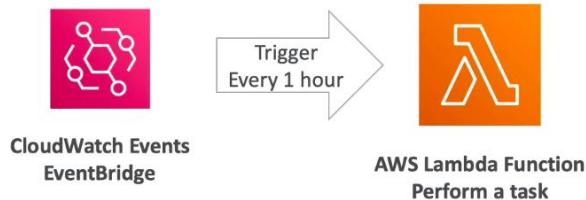
SQS



Cognito



Example: Serverless CRON Job



© Stephane Maarek

DD session

AWS Lambda Pricing: example

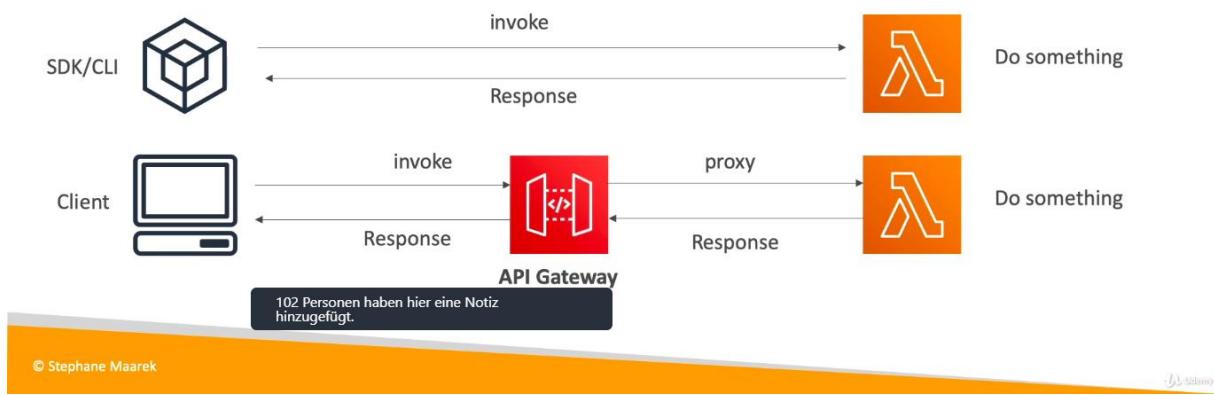
- You can find overall pricing information here:
<https://aws.amazon.com/lambda/pricing/>
- Pay per calls:
 - First 1,000,000 requests are free
 - \$0.20 per 1 million requests thereafter (\$0.0000002 per request)
- Pay per duration: (in increment of 100ms)
 - 400,000 GB-seconds of compute time per month if FREE
 - == 400,000 seconds if function is 1GB RAM
 - == 3,200,000 seconds if function is 128 MB RAM
 - After that \$1.00 for 600,000 GB-seconds
- It is usually very cheap to run AWS Lambda so it's very popular

© Stephane Maarek

DD session

Lambda – Synchronous Invocations

- Synchronous: CLI, SDK, API Gateway, Application Load Balancer
 - Results is returned right away
 - Error handling must happen client side (retries, exponential backoff, etc...)



239. Lambda Synchronous Invocations

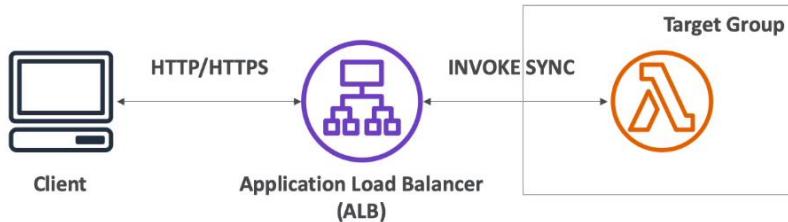
Lambda - Synchronous Invocations - Services

- User Invoked:
 - Elastic Load Balancing (Application Load Balancer)
 - Amazon API Gateway
 - Amazon CloudFront (Lambda@Edge)
 - Amazon S3 Batch
- Service Invoked:
 - Amazon Cognito
 - AWS Step Functions
- Other Services:
 - Amazon Lex
 - Amazon Alexa
 - Amazon Kinesis Data Firehose



Lambda Integration with ALB

- To expose a Lambda function as an HTTP(S) endpoint...
- You can use the Application Load Balancer (or an API Gateway)
- The Lambda function must be registered in a target group



© Stephane Maarek Drucken Sie Esc, um den Vollbildmodus zu verlassen

ALB to Lambda: [HTTP to JSON]

Request Payload for Lambda Function

```
{
  "requestContext": {
    "elb": {
      "targetGroupArn": "arn:aws:elasticloadbalancing:us-east-2:12345678901234567890:targetgroup/Lambda-Target-Group/49e9d65c45c6791a"
    }
  },
  "httpMethod": "GET",
  "path": "/lambda",
  "queryStringParameters": {
    "query": "123ABCD"
  },
  "headers": {
    "connection": "keep-alive",
    "host": "lambda-alb-123578498.us-east-2.elb.amazonaws.com",
    "user-agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.147 Safari/537.36",
    "x-amzn-trace-id": "Root=1-5c536348-3d683b8b04734faae651f476",
    "x-forwarded-for": "72.12.164.125",
    "x-forwarded-port": "80",
    "x-forwarded-proto": "http"
  },
  "body": "",
  "isBase64Encoded": false
}
```

ELB information

HTTP Method & Path

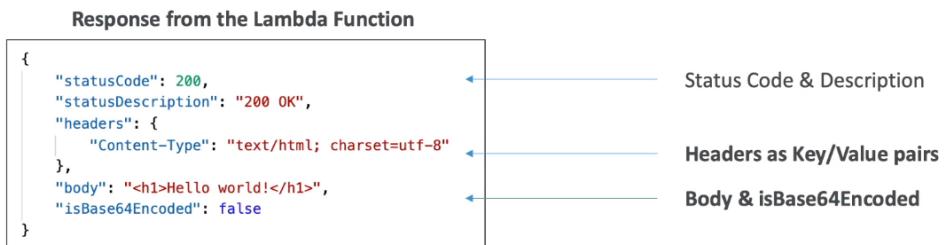
Query String Parameters as Key/Value pairs

Headers as Key/Value pairs

Body (for POST, PUT...) & isBase64Encoded

© Stephane Maarek Drucken Sie Esc, um den Vollbildmodus zu verlassen

Lambda to ALB conversions: JSON to HTTP



© Stephane Maarek

241. Lambda & Application Load Balancer

ALB Multi-Header Values

- ALB can support multi header values (ALB setting)
- When you enable multi-value headers, HTTP headers and query string parameters that are sent with multiple values are shown as arrays within the AWS Lambda event and response objects.

Client

HTTP

http://example.com/path?name=foo&name=bar

ALB

Lambda

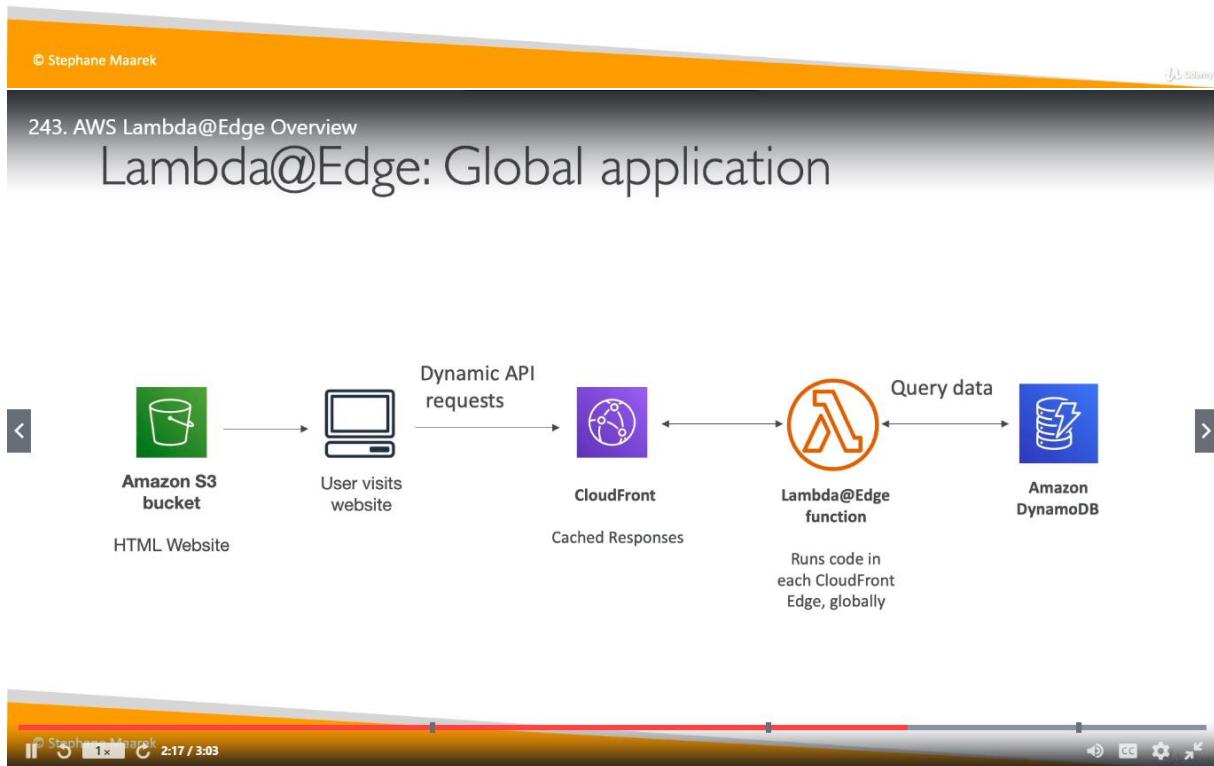
queryStringParameters": {"name": ["foo", "bar"] }

Lambda@Edge

- You can use Lambda to change CloudFront requests and responses:
 - After CloudFront receives a request from a viewer (viewer request)
 - Before CloudFront forwards the request to the origin (origin request)
 - After CloudFront receives the response from the origin (origin response)
 - Before CloudFront forwards the response to the viewer (viewer response)



- You can also generate responses to viewers without ever sending the request to the origin.



Lambda@Edge: Use Cases

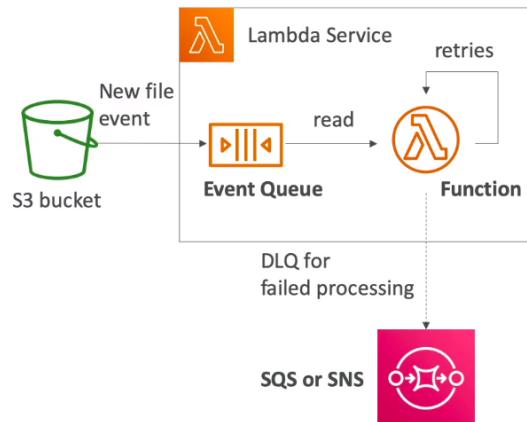
- Website Security and Privacy
- Dynamic Web Application at the Edge
- Search Engine Optimization (SEO)
- Intelligently Route Across Origins and Data Centers
- Bot Mitigation at the Edge
- Real-time Image Transformation
- A/B Testing
- User Authentication and Authorization
- User Prioritization
- User Tracking and Analytics

© Stephane Maarek

DD 30min

Lambda – Asynchronous Invocations

- S3, SNS, CloudWatch Events...
- The events are placed in an Event Queue
- Lambda attempts to retry on errors
 - 3 tries total
 - 1 minute wait after 1st, then 2 minutes wait
- Make sure the processing is idempotent (in case of retries)
- If the function is retried, you will see duplicate logs entries in CloudWatch Logs
- Can define a DLQ (dead-letter queue) – SNS or SQS – for failed processing (need correct IAM permissions)
- Asynchronous invocations allow you to speed up the processing if you don't need to wait for the result (ex: you need 1000 files processed)



© Stephane Maarek

DD 30min

Lambda - Asynchronous Invocations - Services

- Amazon Simple Storage Service (S3)
- Amazon Simple Notification Service (SNS)
- Amazon CloudWatch Events / EventBridge
- AWS CodeCommit (CodeCommit Trigger: new branch, new tag, new push)
- AWS CodePipeline (invoke a Lambda function during the pipeline, Lambda must callback)
- other -----
- Amazon CloudWatch Logs (log processing)
- Amazon Simple Email Service
- AWS CloudFormation
- AWS Config
- AWS IoT
- AWS IoT Events



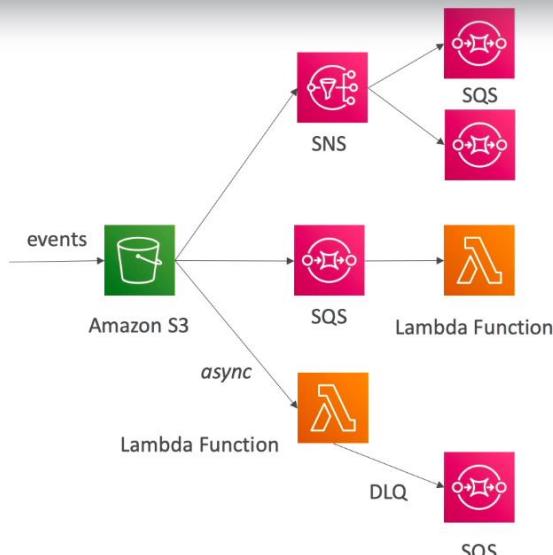
CloudWatch Events / EventBridge



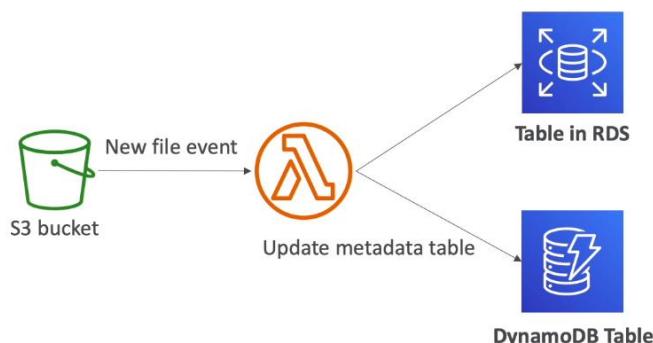
247. Lambda & S3 Event Notifications

- # S3 Events Notifications
- S3:ObjectCreated, S3:ObjectRemoved, S3:ObjectRestore, S3:Replication...
 - Object name filtering possible (*.jpg)
 - Use case: generate thumbnails of images uploaded to S3

 - S3 event notifications typically deliver events in seconds but can sometimes take a minute or longer
 - If two writes are made to a single non-versioned object at the same time, it is possible that only a single event notification will be sent
 - If you want to ensure that an event notification is sent for every successful write, you can enable versioning on your bucket.

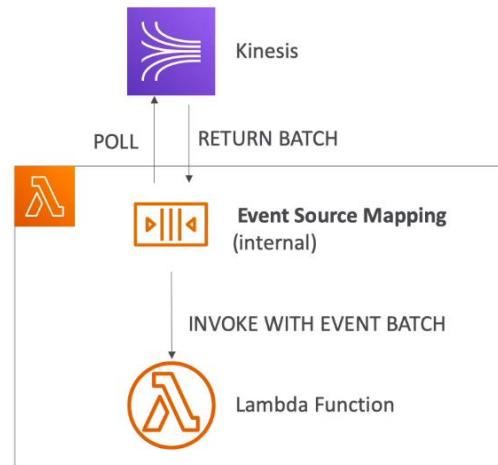


Simple S3 Event Pattern – Metadata Sync



Lambda – Event Source Mapping

- Kinesis Data Streams
- SQS & SQS FIFO queue
- DynamoDB Streams
- Common denominator: records need to be polled from the source
- Your Lambda function is invoked synchronously



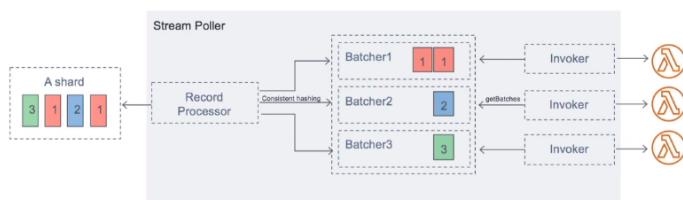
© Stephane Maarek

D3 scaling

248. Lambda Event Source Mapping

Streams & Lambda (Kinesis & DynamoDB)

- An event source mapping creates an iterator for each shard, processes items in order
- Start with new items, from the beginning or from timestamp
- Processed items aren't removed from the stream (other consumers can read them)
- Low traffic: use batch window to accumulate records before processing
- You can process multiple batches in parallel
 - up to 10 batches per shard
 - in-order processing is still guaranteed for each partition key,



<https://aws.amazon.com/blogs/compute/new-aws-lambda-scaling-controls-for-kinesis-and-dynamodb-event-sources/>

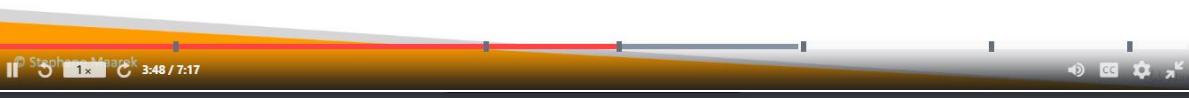
© Stephane Maarek 249 / 717

CC BY-SA

248. Lambda Event Source Mapping

Streams & Lambda – Error Handling

- By default, if your function returns an error, the entire batch is reprocessed until the function succeeds, or the items in the batch expire.
- To ensure in-order processing, processing for the affected shard is paused until the error is resolved
- You can configure the event source mapping to:
 - discard old events
 - restrict the number of retries
 - split the batch on error (to work around Lambda timeout issues)
- Discarded events can go to a Destination



Lambda – Event Source Mapping SQS & SQS FIFO

- Event Source Mapping will poll SQS (Long Polling)
- Specify batch size (1-10 messages)
- Recommended: Set the queue visibility timeout to 6x the timeout of your Lambda function
- To use a DLQ
 - set-up on the SQS queue, not Lambda (DLQ for Lambda is only for async invocations)
 - Or use a Lambda destination for failures

Trigger configuration

SQS

SQS queue

Choose an SQS queue to read messages from.

am:aws:sqsus-east-1:160803060715:new fifo

Batch size

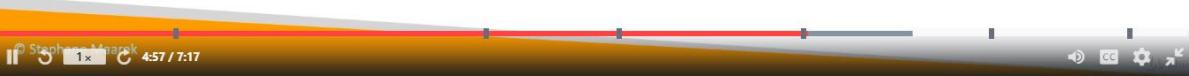
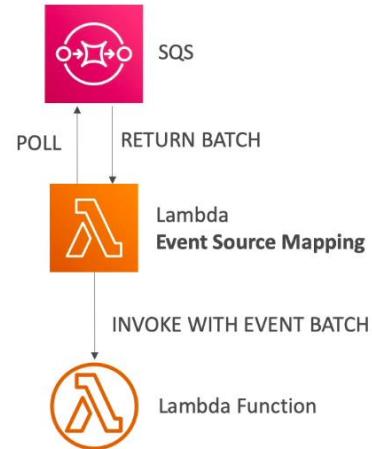
The maximum number of messages to retrieve in a single batch.

10

In order to read from the SQS trigger, your execution role must have proper permissions.

Enable trigger

Enable the trigger now, or create it in a disabled state for testing (recommended).

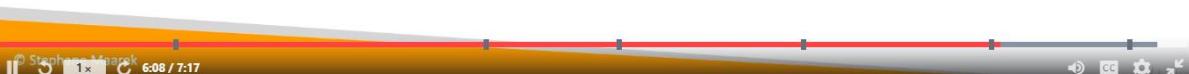


248. Lambda Event Source Mapping

Queues & Lambda

- Lambda also supports in-order processing for FIFO (first-in, first-out) queues, scaling up to the number of active message groups.
- For standard queues, items aren't necessarily processed in order.
- Lambda scales up to process a standard queue as quickly as possible.

- < >
- When an error occurs, batches are returned to the queue as individual items and might be processed in a different grouping than the original batch.
 - Occasionally, the event source mapping might receive the same item from the queue twice, even if no function error occurred.
 - Lambda deletes items from the queue after they're processed successfully.
 - You can configure the source queue to send items to a dead-letter queue if they can't be processed.



248. Lambda Event Source Mapping

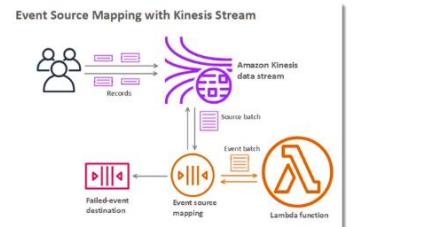
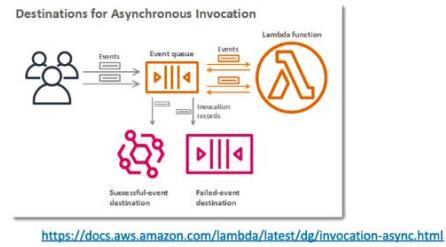
Lambda Event Mapper Scaling

- Kinesis Data Streams & DynamoDB Streams:
 - One Lambda invocation per stream shard
 - If you use parallelization, up to 10 batches processed per shard simultaneously
- SQS Standard:
 - Lambda adds 60 more instances per minute to scale up
 - Up to 1000 batches of messages processed simultaneously
- SQS FIFO:
 - Messages with the same GroupID will be processed in order
 - The Lambda function scales to the number of active message groups



Lambda – Destinations

- Nov 2019: Can configure to send result to a destination
- Asynchronous invocations - can define destinations for successful and failed event:
 - Amazon SQS
 - Amazon SNS
 - AWS Lambda
 - Amazon EventBridge bus
- Note: AWS recommends you use destinations instead of DLQ now (but both can be used at the same time)
- Event Source mapping: for discarded event batches
 - Amazon SQS
 - Amazon SNS
- Note: you can send events to a DLQ directly from SQS



© Stephane Maarek

DD 100%

Lambda Execution Role (IAM Role)



- Grants the Lambda function permissions to AWS services / resources
- Sample managed policies for Lambda:
 - AWSLambdaBasicExecutionRole – Upload logs to CloudWatch.
 - AWSLambdaKinesisExecutionRole – Read from Kinesis
 - AWSLambdaDynamoDBExecutionRole – Read from DynamoDB Streams
 - AWSLambdaSQSQueueExecutionRole – Read from SQS
 - AWSLambdaVPCAccessExecutionRole – Deploy Lambda function in VPC
 - AWSXRayDaemonWriteAccess – Upload trace data to X-Ray.
- When you use an event source mapping to invoke your function, Lambda uses the execution role to read event data.
- Best practice: create one Lambda Execution Role per function

© Stephane Maarek

DD 100%

Lambda Resource Based Policies

- Use resource-based policies to give other accounts and AWS services permission to use your Lambda resources
- Similar to S3 bucket policies for S3 bucket
- An IAM principal can access Lambda:
 - if the IAM policy attached to the principal authorizes it (e.g. user access)
 - OR if the resource-based policy authorizes (e.g. service access)
- When an AWS service like Amazon S3 calls your Lambda function, the resource-based policy gives it access.

© Stephane Maarek

D3 session

Lambda Environment Variables

- Environment variable = key / value pair in "String" form
- Adjust the function behavior without updating code
- The environment variables are available to your code
- Lambda Service adds its own system environment variables as well
- Helpful to store secrets (encrypted by KMS)
- Secrets can be encrypted by the Lambda service key, or your own CMK

© Stephane Maarek

D3 session

Lambda Logging & Monitoring

- CloudWatch Logs:
 - AWS Lambda execution logs are stored in AWS CloudWatch Logs
 - Make sure your AWS Lambda function has an execution role with an IAM policy that authorizes writes to CloudWatch Logs
- CloudWatch Metrics:
 - AWS Lambda metrics are displayed in AWS CloudWatch Metrics
 - Invocations, Durations, Concurrent Executions
 - Error count, Success Rates, Throttles
 - Async Delivery Failures
 - Iterator Age (Kinesis & DynamoDB Streams)

© Stephane Maarek

DD session

Lambda Tracing with X-Ray

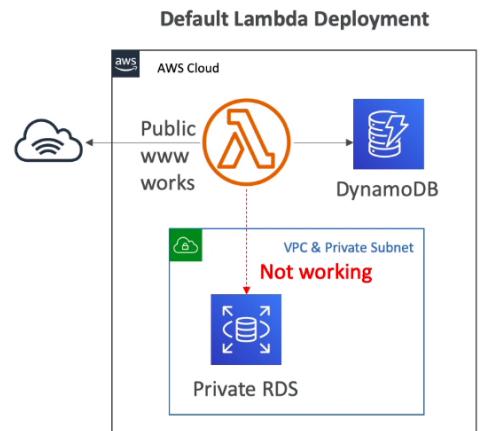
- Enable in Lambda configuration (Active Tracing)
- Runs the X-Ray daemon for you
- Use AWS X-Ray SDK in Code
- Ensure Lambda Function has a correct IAM Execution Role
 - The managed policy is called AWSXRayDaemonWriteAccess
- Environment variables to communicate with X-Ray
 - `_X_AMZN_TRACE_ID`: contains the tracing header
 - `AWS_XRAY_CONTEXT_MISSING`: by default, LOG_ERROR
 - `AWS_XRAY_DAEMON_ADDRESS`: the X-Ray Daemon IP_ADDRESS:PORT

© Stephane Maarek

DD session

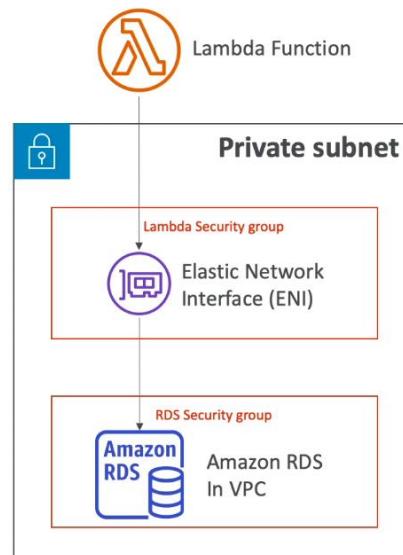
Lambda by default

- By default, your Lambda function is launched outside your own VPC (in an AWS-owned VPC)
- Therefore it cannot access resources in your VPC (RDS, ElastiCache, internal ELB...)



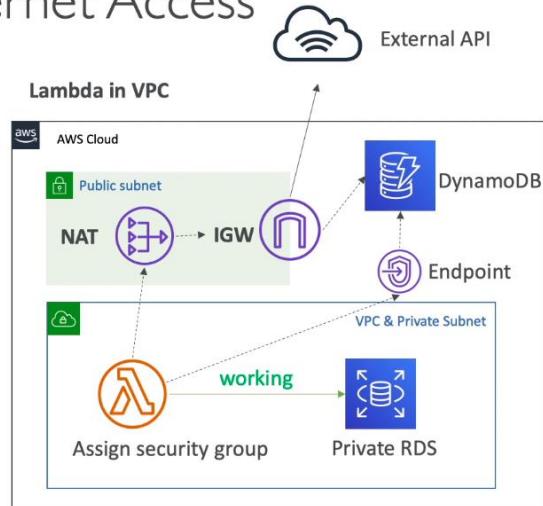
Lambda in VPC

- You must define the VPC ID, the Subnets and the Security Groups
- Lambda will create an ENI (Elastic Network Interface) in your subnets
- AWSLambdaVPCAccessExecutionRole



Lambda in VPC – Internet Access

- A Lambda function in your VPC does not have internet access
- Deploying a Lambda function in a public subnet does not give it internet access or a public IP
- Deploying a Lambda function in a private subnet gives it internet access if you have a NAT Gateway / Instance
- You can use VPC endpoints to privately access AWS services without a NAT



Note: Lambda - CloudWatch Logs works even without endpoint or NAT Gateway

© Stephane Maarek

DD session

256. Lambda Function Performance

Lambda Function Configuration

- RAM:
 - From 128MB to 3,008MB in 64MB increments
 - The more RAM you add, the more vCPU credits you get
 - At 1,792 MB, a function has the equivalent of one full vCPU
 - After 1,792 MB, you get more than one CPU, and need to use multi-threading in your code to benefit from it
- If your application is CPU-bound (computation heavy), increase RAM
- Timeout: default 3 seconds, maximum is 900 seconds (15 minutes)

© Stephane Maarek

DD session

1:42 / 8:59

Lambda Execution Context

- The execution context is a temporary runtime environment that initializes any external dependencies of your lambda code
- Great for database connections, HTTP clients, SDK clients...
- The execution context is maintained for some time in anticipation of another Lambda function invocation
- The next function invocation can “re-use” the context to execution time and save time in initializing connections objects
- The execution context includes the `/tmp` directory



Initialize outside the handler

BAD!

```
import os

def get_user_handler(event, context):

    DB_URL = os.getenv("DB_URL")
    db_client = db.connect(DB_URL)
    user = db_client.get(user_id = event["user_id"])

    return user
```

The DB connection is established
At every function invocation

GOOD!

```
import os

DB_URL = os.getenv("DB_URL")
db_client = db.connect(DB_URL)

def get_user_handler(event, context):

    user = db_client.get(user_id = event["user_id"])

    return user
```

The DB connection is established once
And re-used across invocations



Lambda Functions /tmp space

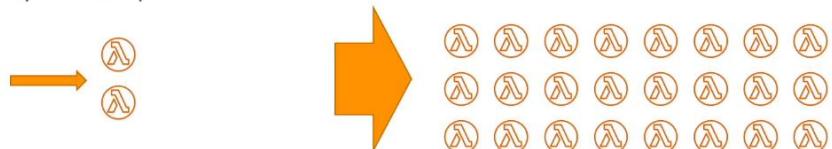
- If your Lambda function needs to download a big file to work...
- If your Lambda function needs disk space to perform operations...
- You can use the /tmp directory
- Max size is 512MB
- The directory content remains when the execution context is frozen, providing transient cache that can be used for multiple invocations (helpful to checkpoint your work)
- For permanent persistence of object (non temporary), use S3

© Stephane Maarek

DD session

Lambda Concurrency and Throttling

- Concurrency limit: up to 1000 concurrent executions



- Can set a “reserved concurrency” at the function level (=limit)
- Each invocation over the concurrency limit will trigger a “Throttle”
- Throttle behavior:
 - If synchronous invocation => return ThrottleError - 429
 - If asynchronous invocation => retry automatically and then go to DLQ
- If you need a higher limit, open a support ticket

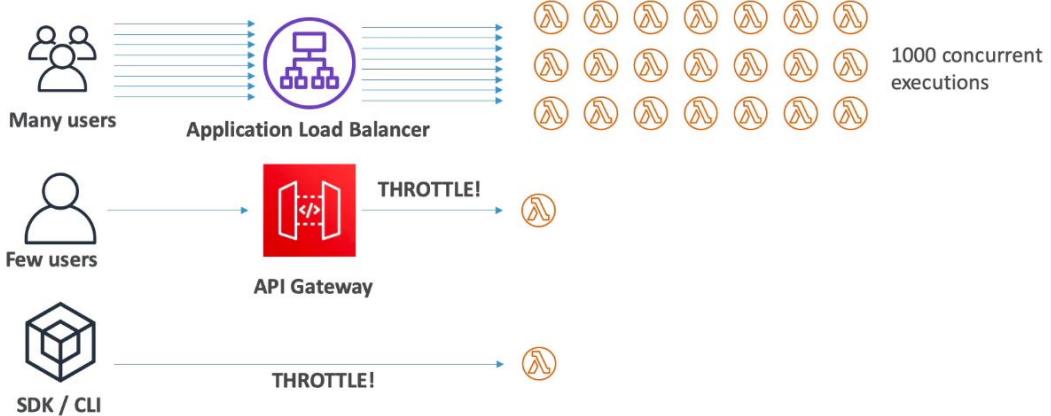
© Stephane Maarek

DD session

257. Lambda Concurrency

Lambda Concurrency Issue

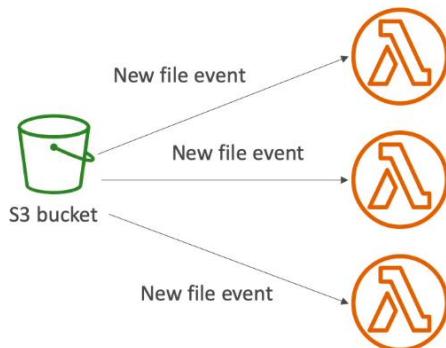
- If you don't reserve (=limit) concurrency, the following can happen:



257. Lambda Concurrency

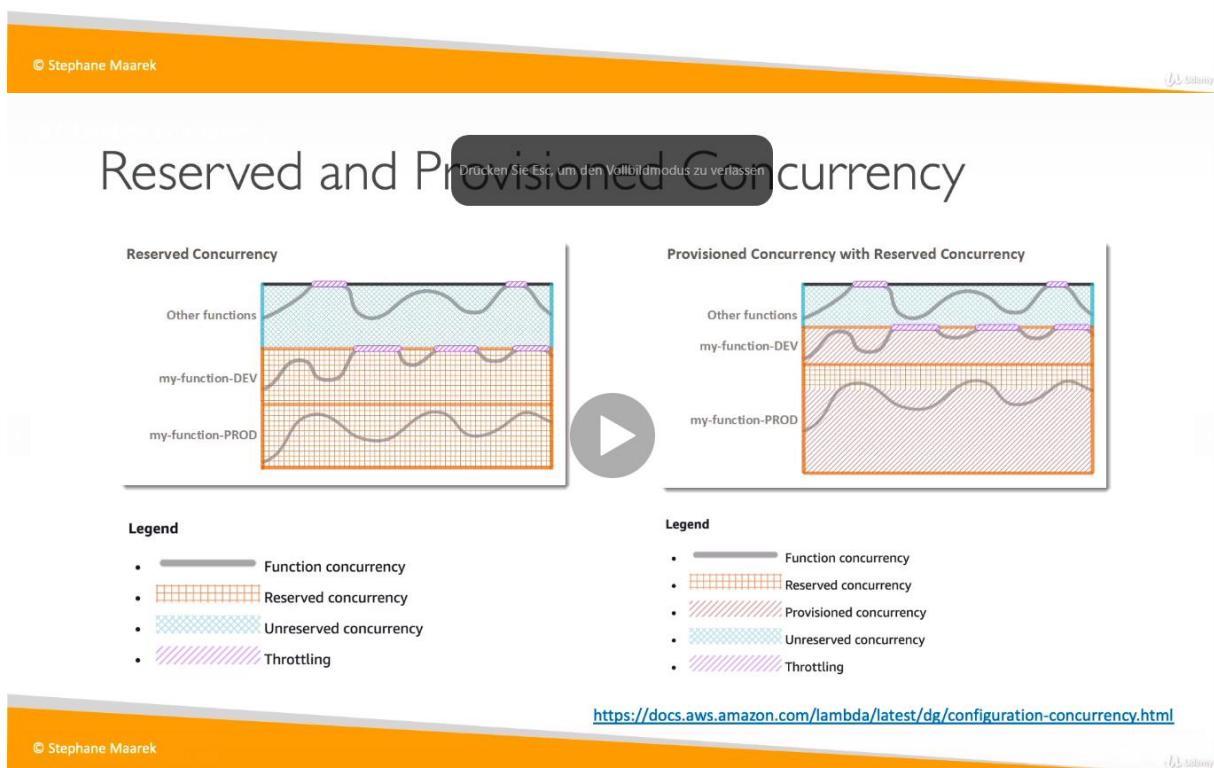
Concurrency and Asynchronous Invocations

- If the function doesn't have enough concurrency available to process all events, additional requests are throttled.
- For throttling errors (429) and system errors (500-series), Lambda returns the event to the queue and attempts to run the function again for up to 6 hours.
- The retry interval increases exponentially from 1 second after the first attempt to a maximum of 5 minutes.



Cold Starts & Provisioned Concurrency

- Cold Start:
 - New instance => code is loaded and code outside the handler run (init)
 - If the init is large (code, dependencies, SDK...) this process can take some time.
 - First request served by new instances has higher latency than the rest
- Provisioned Concurrency:
 - Concurrency is allocated before the function is invoked (in advance)
 - So the cold start never happens and all invocations have low latency
 - Application Auto Scaling can manage concurrency (schedule or target utilization)
- Note:
 - Note: cold starts in VPC have been dramatically reduced in Oct & Nov 2019
 - <https://aws.amazon.com/blogs/compute/announcing-improved-vpc-networking-for-aws-lambda-functions/>



Lambda Function Dependencies

- If your Lambda function depends on external libraries:
for example AWS X-Ray SDK, Database Clients, etc...
- You need to install the packages alongside your code and zip it together
 - For Node.js, use npm & “node_modules” directory
 - For Python, use pip --target options
 - For Java, include the relevant .jar files
- Upload the zip straight to Lambda if less than 50MB, else to S3 first
- Native libraries work: they need to be compiled on Amazon Linux
- AWS SDK comes by default with every Lambda function

The screenshot shows a presentation slide with the following details:

- Title:** 260. Lambda and CloudFormation
- Section:** Lambda and CloudFormation – inline
- Code Block:** A box containing AWS CloudFormation template code. The code defines a Lambda function named "primer" with a Python runtime, a specific role, and a handler. It includes a "ZipFile" section with a sample Python script that connects to a database using the "db" module.
- Right Panel:** A list of bullet points describing inline functions:

- Inline functions are very simple
- Use the **Code.ZipFile** property
- You cannot include function dependencies with inline functions

Lambda and CloudFormation – through S3

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Lambda from S3
Resources:
  Function:
    Type: AWS::Lambda::Function
    Properties:
      Handler: index.handler
      Role: arn:aws:iam::123456789012:role/lambda-role
      Code:
        S3Bucket: my-bucket
        S3Key: function.zip
        S3ObjectVersion: String
      Runtime: nodejs12.x
```

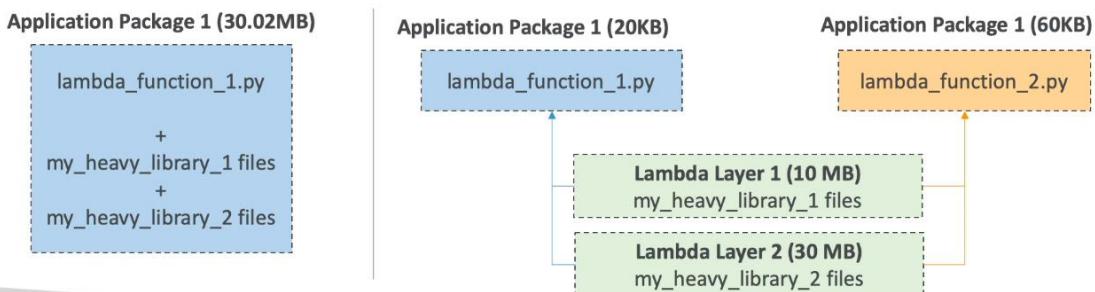
- You must store the Lambda zip in S3
- You must refer the S3 zip location in the CloudFormation code
 - S3Bucket
 - S3Key: full path to zip
 - S3ObjectVersion: if versioned bucket
- If you update the code in S3, but don't update S3Bucket, S3Key or S3ObjectVersion, CloudFormation won't update your function

© Stephane Maarek

DD 30min

Lambda Layers

- Custom Runtimes
 - Ex: C++ <https://github.com/awslabs/aws-lambda-cpp>
 - Ex: Rust <https://github.com/awslabs/aws-lambda-rust-runtime>
- Externalize Dependencies to re-use them:

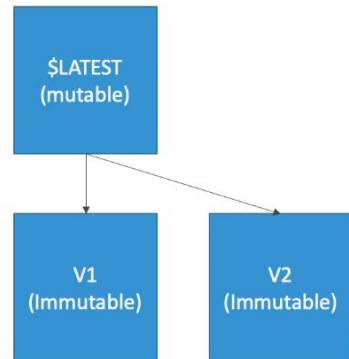


© Stephane Maarek

DD 30min

AWS Lambda Versions

- When you work on a Lambda function, we work on `$LATEST`
- When we're ready to publish a Lambda function, we create a version
- Versions are immutable
- Versions have increasing version numbers
- Versions get their own ARN (Amazon Resource Name)
- Version = code + configuration (nothing can be changed - immutable)
- Each version of the lambda function can be accessed

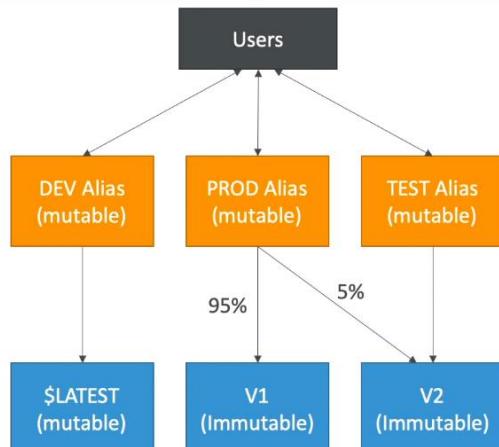


© Stephane Maarek

DD session

262. Lambda Versions and Aliases

- ## AWS Lambda Aliases
- Aliases are "pointers" to Lambda function versions
 - We can define a "dev", "test", "prod" aliases and have them point at different lambda versions
 - Aliases are mutable
 - Aliases enable Blue / Green deployment by assigning weights to lambda functions
 - Aliases enable stable configuration of our event triggers / destinations
 - Aliases have their own ARNs
 - Aliases cannot reference aliases



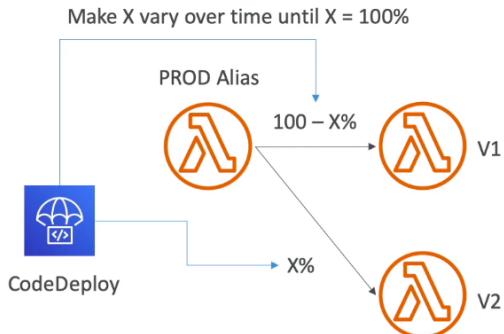
© Stephane Maarek

DD session

3:00 / 8:06

Lambda & CodeDeploy

- CodeDeploy can help you automate traffic shift for Lambda aliases
- Feature is integrated within the SAM framework
- **Linear:** grow traffic every N minutes until 100%
 - Linear|0PercentEvery3Minutes
 - Linear|0PercentEvery10Minutes
- **Canary:** try X percent then 100%
 - Canary|0Percent5Minutes
 - Canary|0Percent30Minutes
- **AllAtOnce:** immediate
- Can create Pre & Post Traffic hooks to check the health of the Lambda function



© Stephane Maarek

33/300

AWS Lambda Limits to Know - per region

- **Execution:**
 - Memory allocation: 128 MB – 3008 MB (64 MB increments)
 - Maximum execution time: 900 seconds (15 minutes)
 - Environment variables (4 KB)
 - Disk capacity in the “function container” (in /tmp): 512 MB
 - Concurrency executions: 1000 (can be increased)
- **Deployment:**
 - Lambda function deployment size (compressed .zip): 50 MB
 - Size of uncompressed deployment (code + dependencies): 250 MB
 - Can use the /tmp directory to load other files at startup
 - Size of environment variables: 4 KB

© Stephane Maarek

34/300

AWS Lambda Best Practices



- Perform heavy-duty work outside of your function handler
 - Connect to databases outside of your function handler
 - Initialize the AWS SDK outside of your function handler
 - Pull in dependencies or datasets outside of your function handler
- Use environment variables for:
 - Database Connection Strings, S3 bucket, etc... don't put these values in your code
 - Passwords, sensitive values... they can be encrypted using KMS
- Minimize your deployment package size to its runtime necessities.
 - Break down the function if need be
 - Remember the AWS Lambda limits
 - Use Layers where necessary
- Avoid using recursive code, never have a Lambda function call itself

© Stephane Maarek

DD session

Traditional Architecture



- Traditional applications leverage RDBMS databases
- These databases have the SQL query language
- Strong requirements about how the data should be modeled
- Ability to do join, aggregations, computations
- Vertical scaling (means usually getting a more powerful CPU / RAM / IO)

AWS Certified Developer © Stephane Maarek

DD session

NoSQL databases

- NoSQL databases are non-relational databases and are **distributed**
 - NoSQL databases include MongoDB, DynamoDB, etc.
 - NoSQL databases do not support join
 - All the data that is needed for a query is present in one row
 - NoSQL databases don't perform aggregations such as "SUM"
 - **NoSQL databases scale horizontally**
-
- There's no "right or wrong" for NoSQL vs SQL, they just require to model the data differently and think about user queries differently

DynamoDB



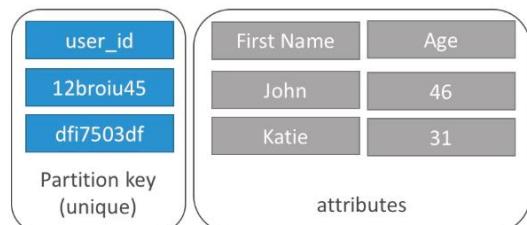
- Fully Managed, Highly available with replication across 3 AZ
- NoSQL database - not a relational database
- Scales to massive workloads, distributed database
- Millions of requests per seconds, trillions of row, 100s of TB of storage
- Fast and consistent in performance (low latency on retrieval)
- Integrated with IAM for security, authorization and administration
- Enables event driven programming with DynamoDB Streams
- Low cost and auto scaling capabilities

DynamoDB - Basics

- DynamoDB is made of **tables**
- Each table has a **primary key** (must be decided at creation time)
- Each table can have an infinite number of items (= rows)
- Each item has **attributes** (can be added over time – can be null)
- Maximum size of a item is 400KB
- Data types supported are:
 - Scalar Types: String, Number, Binary, Boolean, Null
 - Document Types: List, Map
 - Set Types: String Set, Number Set, Binary Set

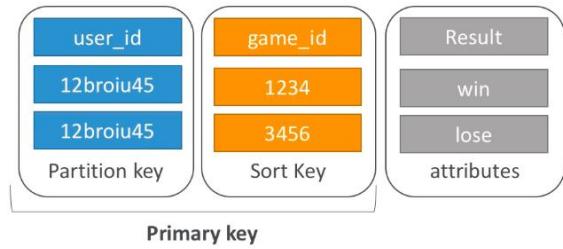
DynamoDB – Primary Keys

- **Option 1: Partition key only (HASH)**
- Partition key must be unique for each item
- Partition key must be “diverse” so that the data is distributed
- Example: user_id for a users table



DynamoDB – Primary Keys

- Option 2: Partition key + Sort Key
- The combination must be unique
- Data is grouped by partition key
- Sort key == range key
- Example: users-games table
 - user_id for the partition key
 - game_id for the sort key



DynamoDB – Partition Keys exercise

- We're building a movie database
- What is the best partition key to maximize data distribution?
 - movie_id
 - producer_name
 - leader_actor_name
 - movie_language
- movie_id has the highest cardinality so it's a good candidate
- movie_language doesn't take many values and may be skewed towards English so it's not a great partition key

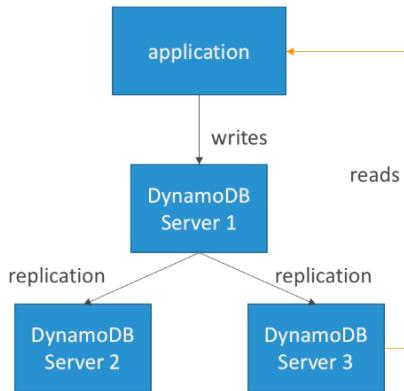
DynamoDB – Provisioned Throughput

- Table must have provisioned read and write capacity units
- **Read Capacity Units (RCU)**: throughput for reads
- **Write Capacity Units (WCU)**: throughput for writes
- Option to setup auto-scaling of throughput to meet demand
- Throughput can be exceeded temporarily using “burst credit”
- If burst credit are empty, you’ll get a “ProvisionedThroughputException”.
- It’s then advised to do an exponential back-off retry

DynamoDB – Write Capacity Units

- One *write capacity unit* represents one write per second for an item up to 1 KB in size.
 - If the items are larger than 1 KB, more WCU are consumed
-
- **Example 1:** we write 10 objects per seconds of 2 KB each.
 - We need $2 * 10 = 20$ WCU
 - **Example 2:** we write 6 objects per second of 4.5 KB each
 - We need $6 * 5 = 30$ WCU (4.5 gets rounded to the upper KB)
 - **Example 3:** we write 120 objects per minute of 2 KB each
 - We need $120 / 60 * 2 = 4$ WCU

Strongly Consistent Read vs Eventually Consistent Read



- **Eventually Consistent Read:** If we read just after a write, it's possible we'll get unexpected response because of replication
- **Strongly Consistent Read:** If we read just after a write, we will get the correct data
- **By default:** DynamoDB uses Eventually Consistent Reads, but GetItem, Query & Scan provide a "ConsistentRead" parameter you can set to True

DynamoDB – Read Capacity Units

- One *read capacity unit* represents one strongly consistent read per second, or two eventually consistent reads per second, for an item up to 4 KB in size.
- If the items are larger than 4 KB, more RCU are consumed
- **Example 1:** 10 strongly consistent reads per seconds of 4 KB each
 - We need $10 * 4 \text{ KB} / 4 \text{ KB} = 10 \text{ RCU}$
- **Example 2:** 16 eventually consistent reads per seconds of 12 KB each
 - We need $(16 / 2) * (12 / 4) = 24 \text{ RCU}$
- **Example 3:** 10 strongly consistent reads per seconds of 6 KB each
 - We need $10 * 8 \text{ KB} / 4 = 20 \text{ RCU}$ (we have to round up 6 KB to 8 KB)

DynamoDB – Partitions Internal

- Data is divided in partitions
- Partition keys go through a hashing algorithm to know to which partition they go to
- To compute the number of partitions:
 - By capacity: $(\text{TOTAL RCU} / 3000) + (\text{TOTAL WCU} / 1000)$
 - By size: Total Size / 10 GB
 - Total partitions = $\text{CEILING}(\text{MAX}(\text{Capacity}, \text{Size}))$
- WCU and RCU are spread evenly between partitions

A screenshot of a presentation slide. The title bar says "269. DynamoDB WCU & RCU - Throughput". The main content is titled "DynamoDB – Partitions Internal". Below the title is a bulleted list of points about partitioning. At the bottom of the slide, there is a navigation bar with icons for back, forward, search, and other presentation controls.

- If we exceed our RCU or WCU, we get **ProvisionedThroughputExceededExceptions**
- Reasons:
 - Hot keys: one partition key is being read too many times (popular item for ex)
 - Hot partitions:
 - Very large items: remember RCU and WCU depends on size of items
- Solutions:
 - Exponential back-off when exception is encountered (already in SDK)
 - Distribute partition keys as much as possible
 - If RCU issue, we can use DynamoDB Accelerator (DAX)

A screenshot of a presentation slide. The title bar says "269. DynamoDB WCU & RCU - Throughput". The main content is titled "DynamoDB - Throttling". Below the title is a bulleted list of points about throttling. At the bottom of the slide, there is a navigation bar with icons for back, forward, search, and other presentation controls.

DynamoDB – Writing Data

- **PutItem** - Write data to DynamoDB (create data or full replace)
 - Consumes WCU
- **UpdateItem** – Update data in DynamoDB (partial update of attributes)
 - Possibility to use Atomic Counters and increase them
- **Conditional Writes:**
 - Accept a write / update only if conditions are respected, otherwise reject
 - Helps with concurrent access to items
 - No performance impact

DynamoDB – Deleting Data

- **DeleteItem**
 - Delete an individual row
 - Ability to perform a conditional delete
- **DeleteTable**
 - Delete a whole table and all its items
 - Much quicker deletion than calling DeleteItem on all items

DynamoDB – Batching Writes

- **BatchWriteItem**
 - Up to 25 **PutItem** and / or **DeleteItem** in one call
 - Up to 16 MB of data written
 - Up to 400 KB of data per item
- Batching allows you to save in latency by reducing the number of API calls done against DynamoDB
- Operations are done in parallel for better efficiency
- It's possible for part of a batch to fail, in which case we have the try the failed items (using exponential back-off algorithm)

AWS Certified Developer © Stephane Maarek

271. DynamoDB Basic APIs

DynamoDB – Reading Data

• **GetItem:**

- Read based on Primary key
- Primary Key = HASH or HASH-RANGE
- Eventually consistent read by default
- Option to use strongly consistent reads (more RCU - might take longer)
- **ProjectionExpression** can be specified to include only certain attributes

• **BatchGetItem:**

- Up to 100 items
- Up to 16 MB of data
- Items are retrieved in parallel to minimize latency

DynamoDB – Query

- **Query** returns items based on:
 - PartitionKey value (**must be = operator**)
 - SortKey value (=, <, <=, >, >=, Between, Begin) – optional
 - FilterExpression to further filter (client side filtering)
- Returns:
 - Up to 1 MB of data
 - Or number of items specified in **Limit**
- Able to do pagination on the results
- Can query table, a local secondary index, or a global secondary index

AWS Certified Developer © Stephane Maarek

DD tutorial

271. DynamoDB Basic APIs

DynamoDB - Scan

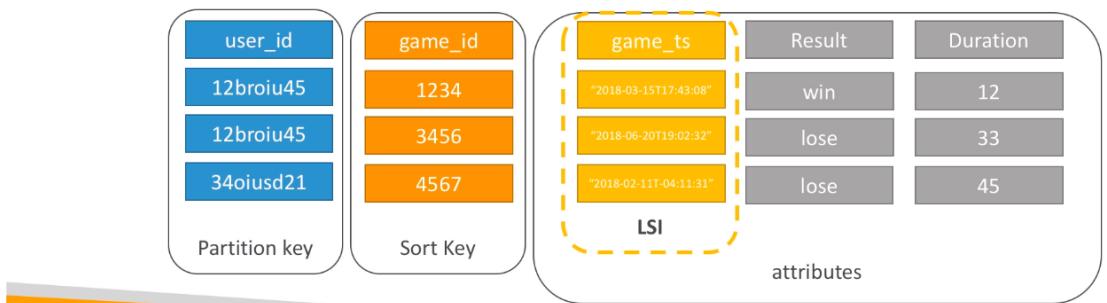
- **Scan** the entire table and then filter out data (inefficient)
- Returns up to 1 MB of data – use pagination to keep on reading
- Consumes a lot of RCU
- Limit impact using **Limit** or reduce the size of the result and pause
- For faster performance, use **parallel scans**:
 - Multiple instances scan multiple partitions at the same time
 - Increases the throughput and RCU consumed
 - Limit the impact of parallel scans just like you would for Scans
- Can use a **ProjectionExpression + FilterExpression** (no change to RCU)

AWS Certified Developer © Stephane Maarek

DD tutorial

DynamoDB – LSI (Local Secondary Index)

- Alternate range key for your table, **local to the hash key**
- Up to five local secondary indexes per table.
- The sort key consists of exactly one scalar attribute.
- The attribute that you choose must be a scalar String, Number, or Binary
- **LSI must be defined at table creation time**



AWS Certified Developer © Stephane Maarek

DD 30min

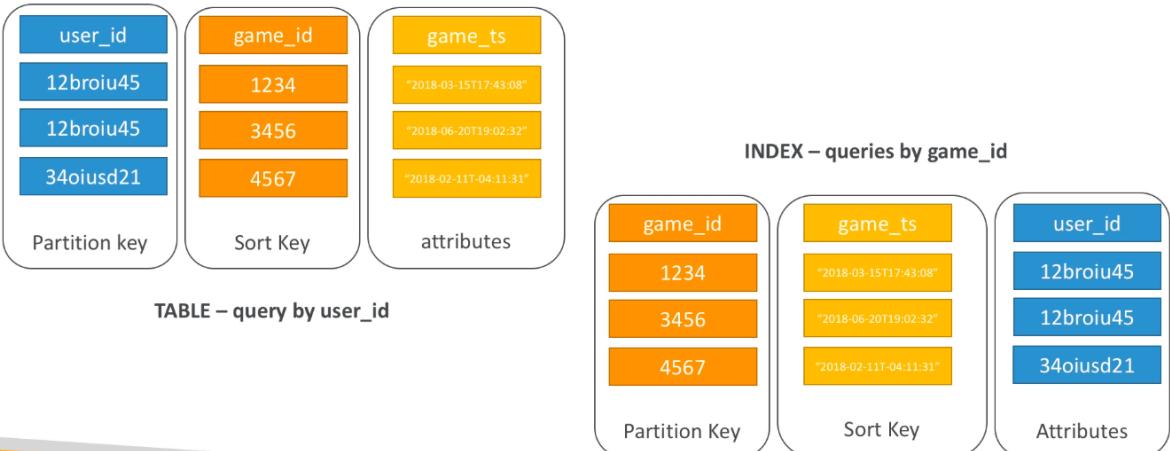
273. DynamoDB Indexes (GSI + LSI)

- ## DynamoDB – GSI (Global Secondary Index)
- To speed up queries on non-key attributes, use a Global Secondary Index
 - GSI = partition key + optional sort key
 - The index is a new “table” and we can project attributes on it
 - The partition key and sort key of the original table are always projected (KEYS_ONLY)
 - Can specify extra attributes to project (INCLUDE)
 - Can use all attributes from main table (ALL)
 - Must define RCU / WCU for the index
 - Possibility to add / modify GSI (not LSI)

AWS Certified Developer © Stephane Maarek

DD 30min

DynamoDB – GSI (Global Secondary Index)



AWS Certified Developer © Stephane Maarek

DD 30m

273. DynamoDB Indexes (GSI + LSI)

DynamoDB Indexes and Throttling

- GSI:

- If the writes are throttled on the GSI, then the main table will be throttled!
- Even if the WCU on the main tables are fine
- Choose your GSI partition key carefully!
- Assign your WCU capacity carefully!



- LSI:

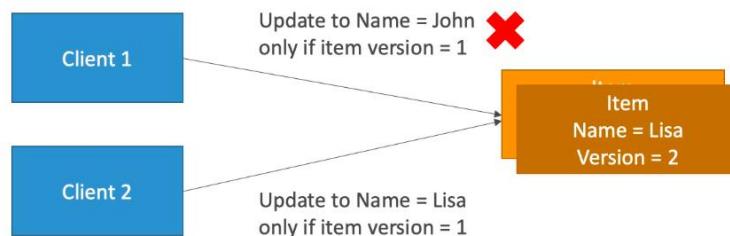
- Uses the WCU and RCU of the main table
- No special throttling considerations

AWS Certified Developer © Stephane Maarek

DD 30m

DynamoDB Concurrency

- DynamoDB has a feature called “Conditional Update / Delete”
- That means that you can ensure an item hasn’t changed before altering it
- That makes DynamoDB an optimistic locking / concurrency database



AWS Certified Developer © Stephane Maarek

275. DynamoDB DAX

DynamoDB - DAX

• DAX = DynamoDB Accelerator

• Seamless cache for DynamoDB, no application re-write

• Writes go through DAX to DynamoDB

• Micro second latency for cached reads & queries

• Solves the Hot Key problem (too many reads)

• 5 minutes TTL for cache by default

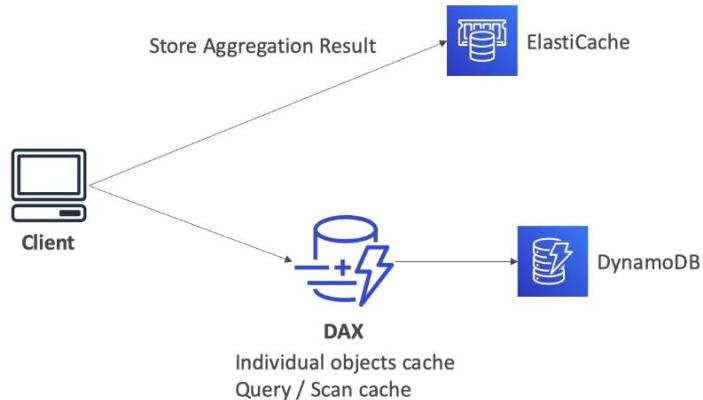
• Up to 10 nodes in the cluster

• Multi AZ (3 nodes minimum recommended for production)

• Secure (Encryption at rest with KMS, VPC, IAM, CloudTrail...)

The diagram illustrates the DAX architecture. At the top, a computer icon represents "applications". Below it is a cylinder icon with a lightning bolt symbol, labeled "DAX". Arrows show bidirectional communication between applications and DAX, and between DAX and three separate "table" icons representing Amazon DynamoDB tables.

DynamoDB – DAX vs ElastiCache



DynamoDB Streams

- Changes in DynamoDB (Create, Update, Delete) can end up in a DynamoDB Stream
- This stream can be read by AWS Lambda & EC2 instances, and we can then do:
 - React to changes in real time (welcome email to new users)
 - Analytics
 - Create derivative tables / views
 - Insert into ElasticSearch
- Could implement cross region replication using Streams
- Stream has 24 hours of data retention



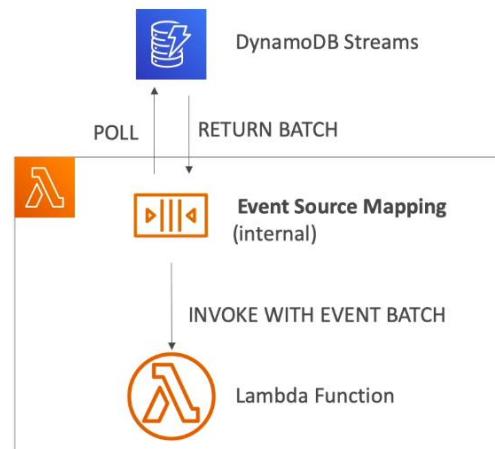
DynamoDB Streams

- Choose the information that will be written to the stream whenever the data in the table is modified:
 - KEYS_ONLY — Only the key attributes of the modified item.
 - NEW_IMAGE — The entire item, as it appears after it was modified.
 - OLD_IMAGE — The entire item, as it appeared before it was modified.
 - NEW_AND_OLD_IMAGES — Both the new and the old images of the item.
- DynamoDB Streams are made of shards, just like Kinesis Data Streams
- You don't provision shards, this is automated by AWS
- Records are not retroactively populated in a stream after enabling it

276. DynamoDB Streams

DynamoDB Streams & Lambda

- You need to define an Event Source Mapping to read from a DynamoDB Streams
- You need to ensure the Lambda function has the appropriate permissions
- Your Lambda function is invoked synchronously



DynamoDB - TTL (Time to Live)

- TTL = automatically delete an item after an expiry date / time
- TTL is provided at no extra cost, deletions do not use WCU / RCU
- TTL is a background task operated by the DynamoDB service itself
- Helps reduce storage and manage the table size over time
- Helps adhere to regulatory norms
- TTL is enabled per row (you define a TTL column, and add a date there)
- DynamoDB typically deletes expired items within 48 hours of expiration
- Deleted items due to TTL are also deleted in GSI / LSI
- DynamoDB Streams can help recover expired items

DynamoDB CLI – Good to Know

- --projection-expression : attributes to retrieve
- --filter-expression : filter results
- General CLI pagination options including DynamoDB / S3:
 - Optimization:
 - --page-size : full dataset is still received but each API call will request less data (helps avoid timeouts)
 - Pagination:
 - --max-items : max number of results returned by the CLI. Returns *NextToken*
 - --starting-token: specify the last received *NextToken* to keep on reading

DynamoDB Transactions

- New feature from November 2018
- Transaction = Ability to Create / Update / Delete multiple rows in different tables at the same time
- It's an "all or nothing" type of operation.
- Write Modes: Standard, Transactional
- Read Modes: Eventual Consistency, Strong Consistency, Transactional
- Consume 2x of WCU / RCU

AWS Certified Developer © Stephane Maarek

279. DynamoDB Transactions

DynamoDB Transactions

- AccountBalance Table

Account_id	balance	Last_transaction_ts
Acct_21	230	1562503085
Acct_45	120	1562503085

- BankTransactions Table

Transaction_id	Transaction_time	From_account_id	To_account_id	value
Tx_12345	1561483349	Acct_45	Acct_21	45
Tx_23456	1562503085	Acct_21	Acct_45	100

- A transaction is a write to both table, or none!

DynamoDB as Session State Cache

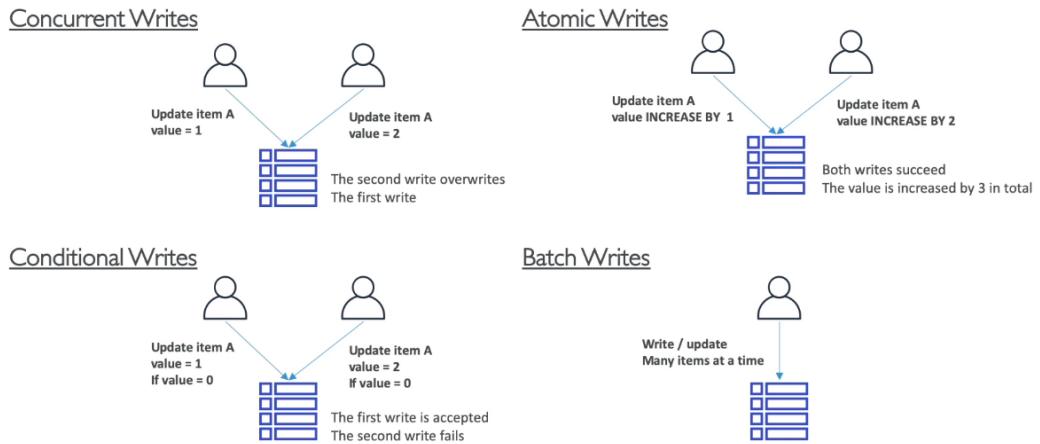
- It's common to use DynamoDB to store session state
- vs ElastiCache:
 - ElastiCache is in-memory, but DynamoDB is serverless
 - Both are key/value stores
- vs EFS:
 - EFS must be attached to EC2 instances as a network drive
- vs EBS & Instance Store:
 - EBS & Instance Store can only be used for local caching, not shared caching
- vs S3:
 - S3 is higher latency, and not meant for small objects

The screenshot shows a presentation slide with the following details:

- Title:** 281. DynamoDB Partitioning Strategies
- Section:** DynamoDB Write Sharding
- Content:**
 - Imagine we have a voting application with two candidates, candidate A and candidate B.
 - If we use a partition key of candidate_id, we will run into partitions issues, as we only have two partitions
 - Solution: add a suffix (usually random suffix, sometimes calculated suffix)
- Table:** A table illustrating the solution. It has three columns: Partition Key, Sort_Key, and Attributes. The data is as follows:

Partition Key	Sort_Key	Attributes
CandidateID + RandomSuffix	Vote_date	Voter_id
Candidate_A-1	2016-05-17 01.36.45	235343
Candidate_A-1	2016-05-18 01.36.30	232312
Candidate_A-2	2016-06-15 01.36.20	098432
Candidate_B-1	2016-07-1 01.36.15	340983

DynamoDB – Write Types

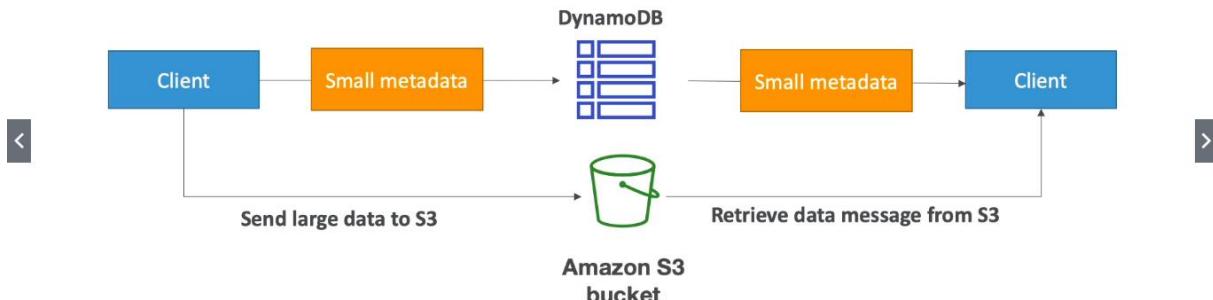


© Stephane Maarek

DynamoDB

283. DynamoDB Patterns with S3

DynamoDB - Large Objects Pattern

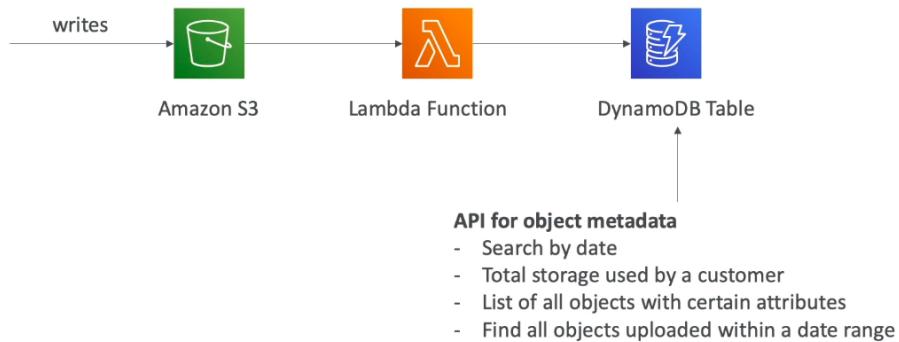


© Stephane Maarek

CC BY-SA

1:11 / 2:16

DynamoDB - Indexing S3 objects metadata



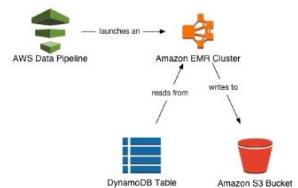
© Stephane Maarek

DD 30min

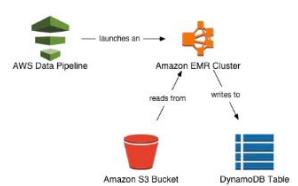
DynamoDB Operations

- **Table Cleanup:**
 - Option 1: Scan + Delete => very slow, expensive, consumes RCU & WCU
 - Option 2: Drop Table + Recreate table => fast, cheap, efficient
- **Copying a DynamoDB Table:**
 - Option 1: Use AWS DataPipeline (uses EMR)
 - Option 2: Create a backup and restore the backup into a new table name (can take some time)
 - Option 3: Scan + Write => write own code

Exporting Data from DynamoDB to Amazon S3



Importing Data from Amazon S3 to DynamoDB



© Stephane Maarek

DD 30min

DynamoDB – Security & Other Features

- Security:
 - VPC Endpoints available to access DynamoDB without internet
 - Access fully controlled by IAM
 - Encryption at rest using KMS
 - Encryption in transit using SSL /TLS
- Backup and Restore feature available
 - Point in time restore like RDS
 - No performance impact
- Global Tables
 - Multi region, fully replicated, high performance
- Amazon DMS can be used to migrate to DynamoDB (from Mongo, Oracle, MySQL, S3, etc...)
- You can launch a local DynamoDB on your computer for development purposes

Example: Building a Serverless API



AWS API Gateway



- AWS Lambda + API Gateway: No infrastructure to manage
- Support for the WebSocket Protocol
- Handle API versioning (v1, v2...)
- Handle different environments (dev, test, prod...)
- Handle security (Authentication and Authorization)
- Create API keys, handle request throttling
- Swagger / Open API import to quickly define APIs
- Transform and validate requests and responses
- Generate SDK and API specifications
- Cache API responses

API Gateway – Integrations High Level

- Lambda Function
 - Invoke Lambda function
 - Easy way to expose REST API backed by AWS Lambda
- HTTP
 - Expose HTTP endpoints in the backend
 - Example: internal HTTP API on premise, Application Load Balancer...
 - Why? Add rate limiting, caching, user authentications, API keys, etc...
- AWS Service
 - Expose any AWS API through the API Gateway?
 - Example: start an AWS Step Function workflow, post a message to SQS
 - Why? Add authentication, deploy publicly, rate control...

API Gateway - Endpoint Types

- Edge-Optimized (default): For global clients
 - Requests are routed through the CloudFront Edge locations (improves latency)
 - The API Gateway still lives in only one region
- Regional:
 - For clients within the same region
 - Could manually combine with CloudFront (more control over the caching strategies and the distribution)
- Private:
 - Can only be accessed from your VPC using an interface VPC endpoint (ENI)
 - Use a resource policy to define access



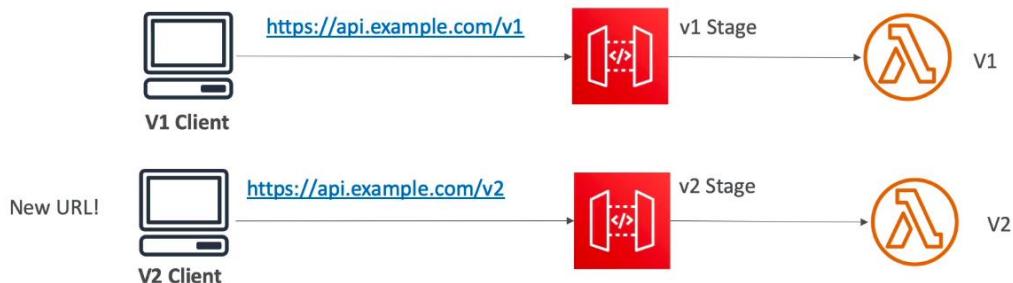
API Gateway – Deployment Stages

- Making changes in the API Gateway does not mean they're effective
- You need to make a "deployment" for them to be in effect
- It's a common source of confusion
- Changes are deployed to "Stages" (as many as you want)
- Use the naming you like for stages (dev, test, prod)
- Each stage has its own configuration parameters
- Stages can be rolled back as a history of deployments is kept



API Gateway – Stages v1 and v2

API breaking change

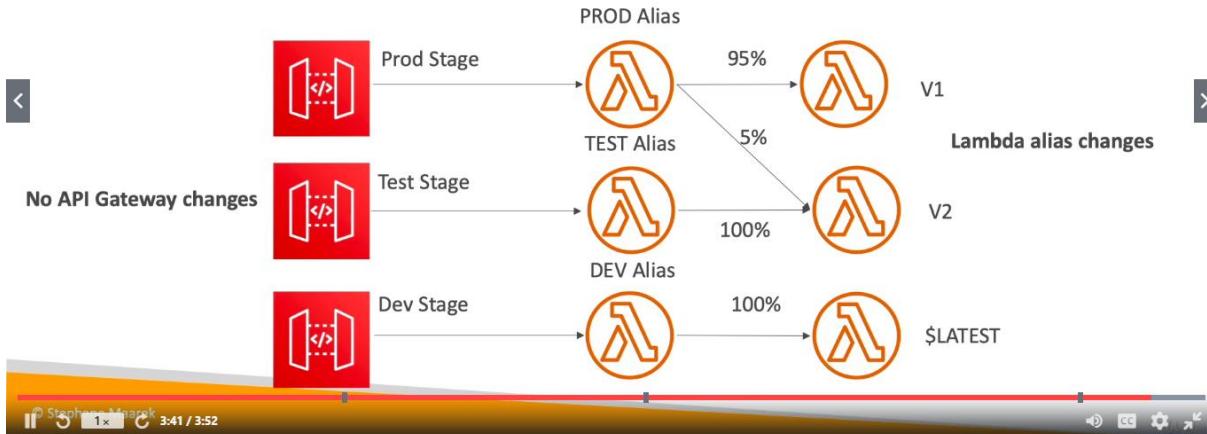


API Gateway – Stage Variables

- Stage variables are like environment variables for API Gateway
- Use them to change often changing configuration values
- They can be used in:
 - Lambda function ARN
 - HTTP Endpoint
 - Parameter mapping templates
- Use cases:
 - Configure HTTP endpoints your stages talk to (dev, test, prod...)
 - Pass configuration parameters to AWS Lambda through mapping templates
- Stage variables are passed to the "context" object in AWS Lambda

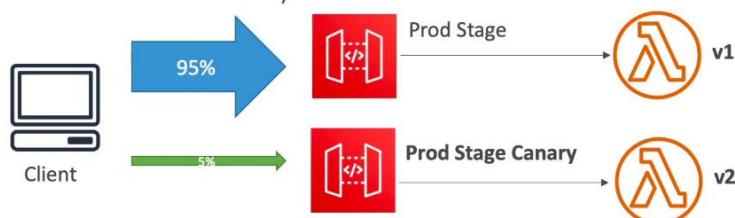
API Gateway Stage Variables & Lambda Aliases

- We create a **stage variable** to indicate the corresponding Lambda alias
- Our API gateway will automatically invoke the right Lambda function!



API Gateway – Canary Deployment

- Possibility to enable canary deployments for any stage (usually prod)
- Choose the % of traffic the canary channel receives



- Metrics & Logs are separate (for better monitoring)
- Possibility to override stage variables for canary
- This is blue / green deployment with AWS Lambda & API Gateway

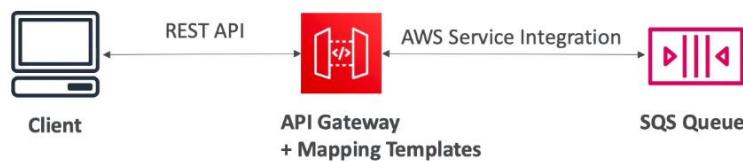
API Gateway - Integration Types

- Integration Type MOCK

- API Gateway returns a response without sending the request to the backend

- Integration Type HTTP / AWS (Lambda & AWS Services)

- you must configure both the integration request and integration response
- Setup data mapping using mapping templates for the request & response



© Stephane Maarek

DD 2019

294. API Gateway Integration Types & Mappings

API Gateway - Integration Types

- Integration Type AWS_PROXY (Lambda Proxy):

- incoming request from the client is the input to Lambda
- The function is responsible for the logic of request / response
- No mapping template, headers, query string parameters... are passed as arguments

```
{
  "resource": "Resource path",
  "path": "Path parameter",
  "httpMethod": "Incoming request's method name",
  "headers": "String containing incoming request headers",
  "multiValueHeaders": "List of strings containing incoming multi-value headers",
  "queryStringParameters": "query string parameters",
  "multiValueQueryStringParameters": "List of query string parameters",
  "pathParameters": "path parameters",
  "stageVariables": "Applicable stage variables",
  "requestContext": "Request context, including authorizer",
  "body": "A JSON string of the request payload.",
  "isBase64Encoded": "A boolean flag"
}
```

Lambda function invocation payload

```
{
  "isBase64Encoded": "true|false",
  "statusCode": "httpStatusCode",
  "headers": { "headerName": "HeaderValue", ... },
  "multiValueHeaders": { "headerName": [ "HeaderValue", "HeaderValue" ] },
  "body": "..."
}
```

Lambda function expected response

© Stephane Maarek 225 / 5:14

API Gateway - Integration Types

- Integration Type **HTTP_PROXY**
 - No mapping template
 - The HTTP request is passed to the backend
 - The HTTP response from the backend is forwarded by API Gateway



© Stephane Maarek

DD 2019

Mapping Templates (AWS & HTTP Integration)

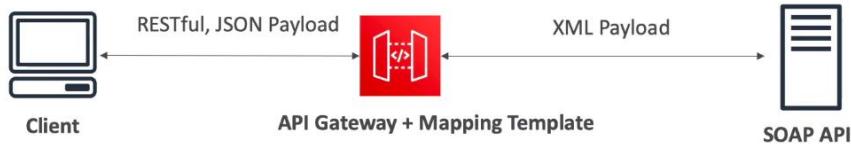
- Mapping templates can be used to modify request / responses
- Rename / Modify query string parameters
- Modify body content
- Add headers
- Uses Velocity Template Language (VTL): for loop, if etc...
- Filter output results (remove unnecessary data)

© Stephane Maarek

DD 2019

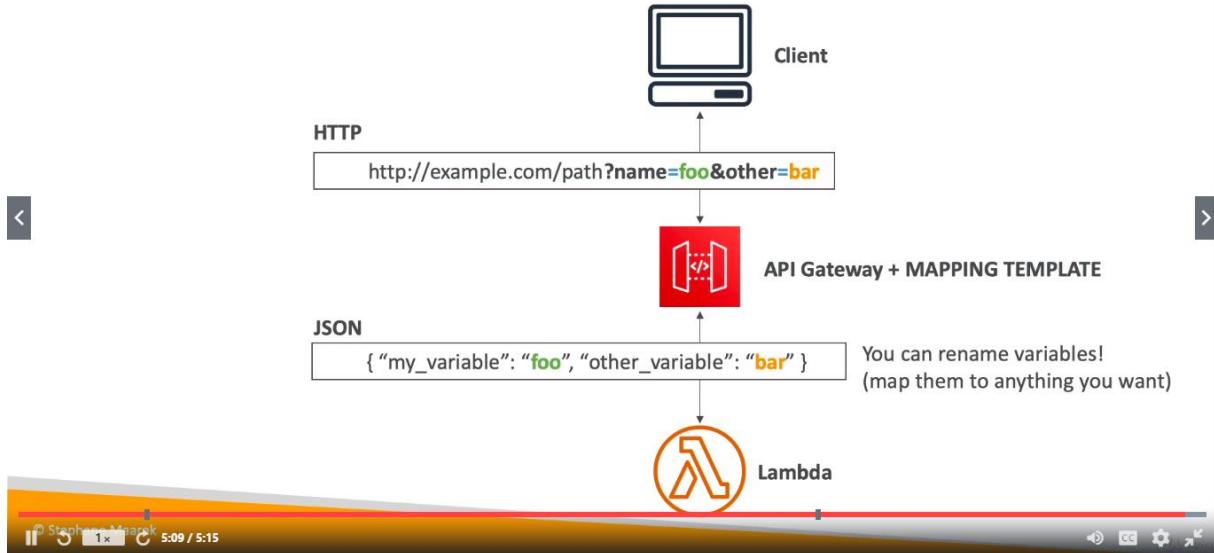
Mapping Example: JSON to XML with SOAP

- SOAP API are XML based, whereas REST API are JSON based



- In this case, API Gateway should:
 - Extract data from the request: either path, payload or header
 - Build SOAP message based on request data (mapping template)
 - Call SOAP service and receive XML response
 - Transform XML response to desired format (like JSON), and respond to the user

Mapping Example: Query String parameters



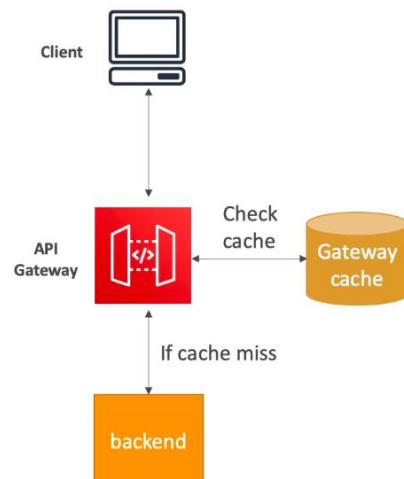
AWS API Gateway Swagger / Open API spec

- Common way of defining REST APIs, using API definition as code
- Import existing Swagger / OpenAPI 3.0 spec to API Gateway
 - Method
 - Method Request
 - Integration Request
 - Method Response
 - + AWS extensions for API gateway and setup every single option
- Can export current API as Swagger / OpenAPI spec
- Swagger can be written in YAML or JSON
- Using Swagger we can generate SDK for our applications



Caching API responses

- Caching reduces the number of calls made to the backend
- Default TTL (time to live) is 300 seconds (min: 0s, max: 3600s)
- Caches are defined per stage
- Possible to override cache settings per method
- Cache encryption option
- Cache capacity between 0.5GB to 237GB
- Cache is expensive, makes sense in production, may not make sense in dev / test



API Gateway Cache Invalidation

- Able to flush the entire cache (invalidate it) immediately
- Clients can invalidate the cache with header: Cache-Control: max-age=0 (with proper IAM authorization)
- If you don't impose an InvalidateCache policy (or choose the Require authorization check box in the console), any client can invalidate the API cache

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "execute-api:InvalidateCache"  
            ],  
            "Resource": [  
                "arn:...:api-id/stage-name/GET/resource-path-specifier"  
            ]  
        }  
    ]  
}
```

© Stephane Maarek

33 / 300

API Gateway – Usage Plans & API Keys

- If you want to make an API available as an offering (\$) to your customers
- **Usage Plan:**
 - who can access one or more deployed API stages and methods
 - how much and how fast they can access them
 - uses API keys to identify API clients and meter access
 - configure throttling limits and quota limits that are enforced on individual client
- **API Keys:**
 - alphanumeric string values to distribute to your customers
 - Ex: WBjHxNtoAb4WPKBC7cGm64CBiblb24b4jt8jjHo9
 - Can use with usage plans to control access
 - Throttling limits are applied to the API keys
 - Quotas limits is the overall number of maximum requests

© Stephane Maarek

33 / 300

API Gateway – Correct Order for API keys

- To configure a usage plan
1. Create one or more APIs, configure the methods to require an API key, and deploy the APIs to stages.
 2. Generate or import API keys to distribute to application developers (your customers) who will be using your API.
 3. Create the usage plan with the desired throttle and quota limits.
 4. Associate API stages and API keys with the usage plan.
- Callers of the API must supply an assigned API key in the x-api-key header in requests to the API.

© Stephane Maarek

D3.js

200 API Gateway – Logging and Tracing

API Gateway – Logging & Tracing

- CloudWatch Logs:
 - Enable CloudWatch logging at the Stage level (with Log Level)
 - Can override settings on a per API basis (ex: ERROR, DEBUG, INFO)
 - Log contains information about request / response body
- X-Ray:
 - Enable tracing to get extra information about requests in API Gateway
 - X-Ray API Gateway + AWS Lambda gives you the full picture

© Stephane Maarek

D3.js

API Gateway – CloudWatch Metrics



- Metrics are by stage, Possibility to enable detailed metrics
- CacheHitCount & CacheMissCount: efficiency of the cache
- Count: The total number API requests in a given period.
- IntegrationLatency: The time between when API Gateway relays a request to the backend and when it receives a response from the backend.
- Latency: The time between when API Gateway receives a request from a client and when it returns a response to the client. The latency includes the integration latency and other API Gateway overhead.
- 4XXError (client-side) & 5XXError (server-side)

© Stephane Maarek

DD session

API Gateway Throttling

- Account Limit
 - API Gateway throttles requests at 10000 rps across all API
 - Soft limit that can be increased upon request
 - In case of throttling => 429 Too Many Requests (retryable error)
 - Can set Stage limit & Method limits to improve performance
 - Or you can define Usage Plans to throttle per customer
- Just like Lambda Concurrency, one API that is overloaded, if not limited, can cause the other APIs to be throttled

© Stephane Maarek

DD session

API Gateway - Errors

- 4xx means Client errors
 - 400: Bad Request
 - 403: Access Denied, WAF filtered
 - 429: Quota exceeded, Throttle
- 5xx means Server errors
 - 502: Bad Gateway Exception, usually for an incompatible output returned from a Lambda proxy integration backend and occasionally for out-of-order invocations due to heavy loads.
 - 503: Service Unavailable Exception
 - 504: Integration Failure – ex Endpoint Request Timed-out Exception
API Gateway requests time out after 29 second maximum

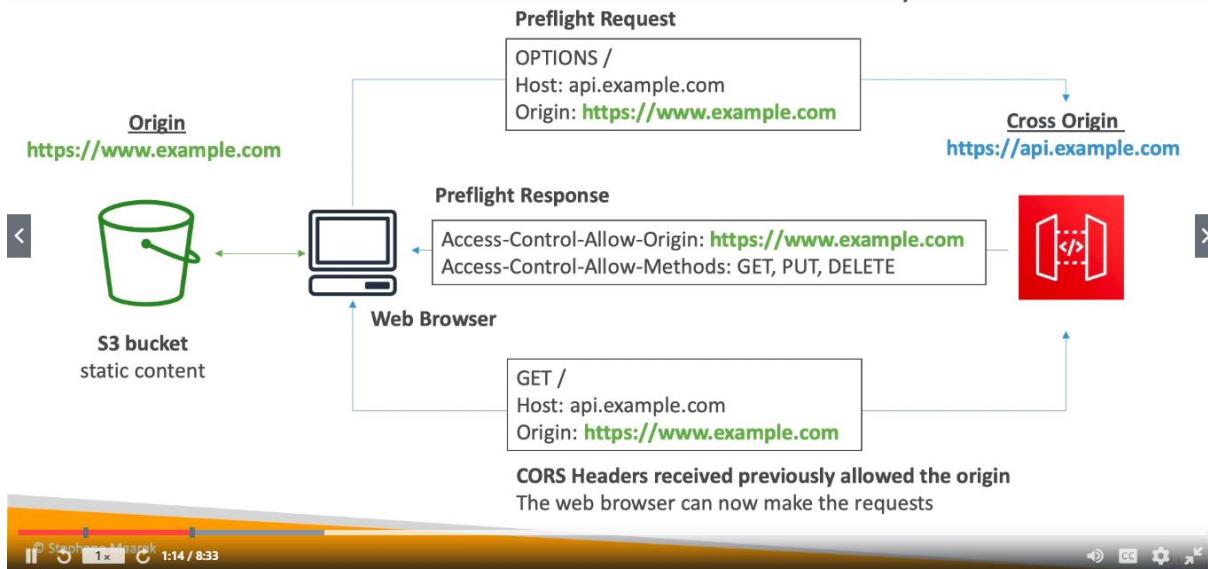


AWS API Gateway - CORS

- CORS must be enabled when you receive API calls from another domain.
- The OPTIONS pre-flight request must contain the following headers:
 - Access-Control-Allow-Methods
 - Access-Control-Allow-Headers
 - Access-Control-Allow-Origin
- CORS can be enabled through the console

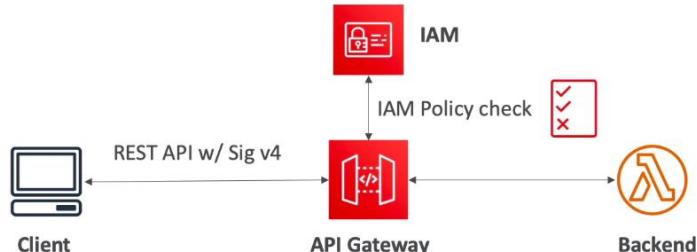


CORS – Enabled on the API Gateway



API Gateway – Security IAM Permissions

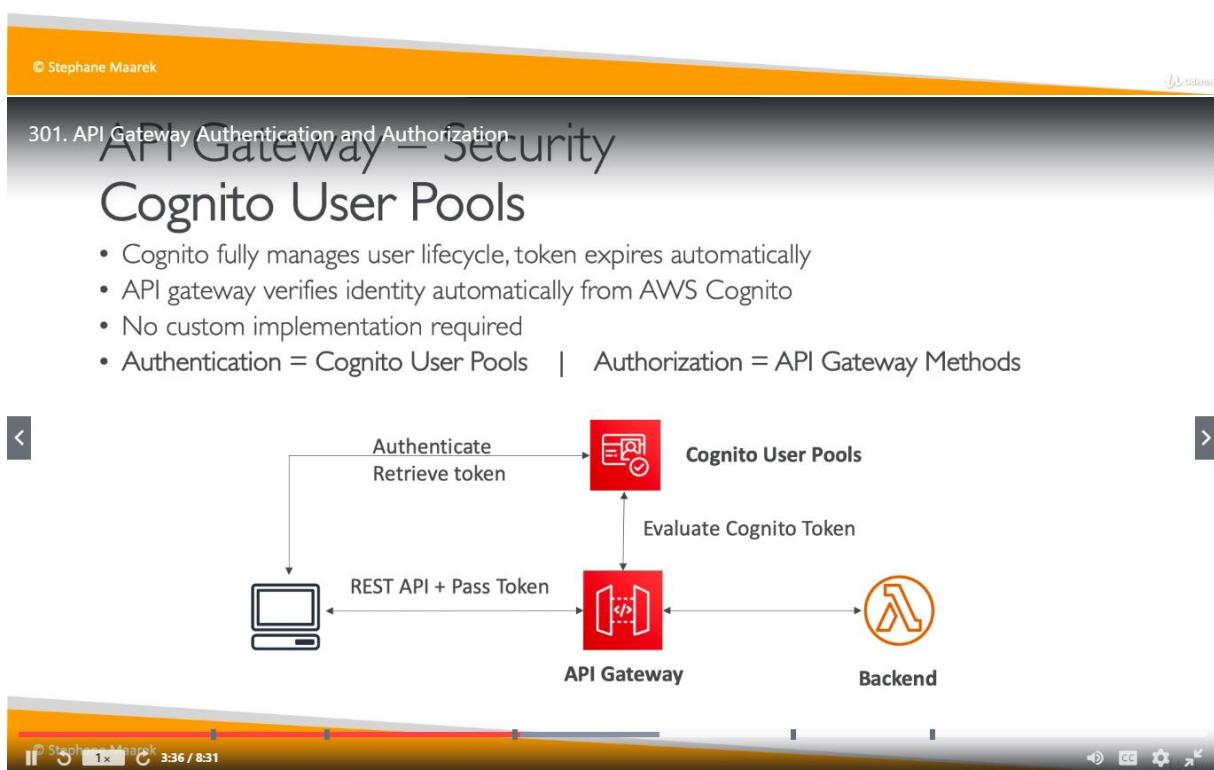
- Create an IAM policy authorization and attach to User / Role
- Authentication = IAM | Authorization = IAM Policy
- Good to provide access within AWS (EC2, Lambda, IAM users...)
- Leverages “Sig v4” capability where IAM credential are in headers



API Gateway – Resource Policies

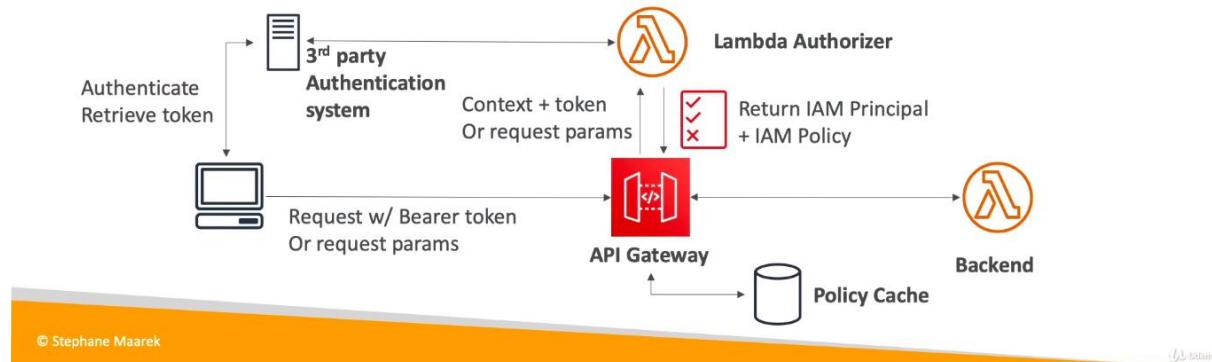
- Resource policies (similar to Lambda Resource Policy)
- Allow for Cross Account Access (combined with IAM Security)
- Allow for a specific source IP address
- Allow for a VPC Endpoint

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": [  
          "arn:aws:iam::account-id-2:user/Alice",  
          "account-id-2"  
        ]  
      },  
      "Action": "execute-api:Invoke",  
      "Resource": [  
        "arn:aws:execute-api:region:account-id-1:api-id/stage/GET/pets"  
      ]  
    }  
  ]  
}
```



API Gateway – Security Lambda Authorizer (formerly Custom Authorizers)

- Token-based authorizer (bearer token) – ex JWT (JSON Web Token) or Oauth
- A request parameter-based Lambda authorizer (headers, query string, stage var)
- Lambda must return an IAM policy for the user; result policy is cached
- Authentication = External | Authorization = Lambda function



API Gateway – Security – Summary

- **IAM:**
 - Great for users / roles already within your AWS account, + resource policy for cross account
 - Handle authentication + authorization
 - Leverages Signature v4
- **Custom Authorizer:**
 - Great for 3rd party tokens
 - Very flexible in terms of what IAM policy is returned
 - Handle Authentication verification + Authorization in the Lambda function
 - Pay per Lambda invocation, results are cached
- **Cognito User Pool:**
 - You manage your own user pool (can be backed by Facebook, Google login etc...)
 - No need to write any custom code
 - Must implement authorization in the backend



API Gateway – HTTP API vs REST API

- **HTTP APIs**

- low-latency, cost-effective AWS Lambda proxy, HTTP proxy APIs and private integration (no data mapping)
- support OIDC and OAuth 2.0 authorization, and built-in support for CORS
- No usage plans and API keys

Authorizers	HTTP API	REST API
AWS Lambda		✓
IAM		✓
Amazon Cognito	✓ *	✓
Native OpenID Connect / OAuth 2.0	✓	

- **REST APIs**

- All features (except Native OpenID Connect / OAuth 2.0)

Full list here: <https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-vs-rest.html>

© Stephane Maarek

DJ Simeon

302. API Gateway REST API vs HTTP API vs WebSocket API

API Gateway – WebSocket API – Overview

- What's WebSocket?

- Two-way interactive communication between a user's browser and a server
- Server can push information to the client
- This enables **stateful** application use cases

- WebSocket APIs are often used in **real-time applications** such as chat applications, collaboration platforms, multiplayer games, and financial trading platforms.

- Works with AWS Services (Lambda, DynamoDB) or HTTP endpoints

© Stephane Maarek

2:17 / 3:01

CC BY NC SA

AWS SAM



- SAM = Serverless Application Model
- Framework for developing and deploying serverless applications
- All the configuration is YAML code
- Generate complex CloudFormation from simple SAM YAML file
- Supports anything from CloudFormation: Outputs, Mappings, Parameters, Resources...
- Only two commands to deploy to AWS
- SAM can use CodeDeploy to deploy Lambda functions
- SAM can help you to run Lambda, API Gateway, DynamoDB locally

AWS Certified Developer © Stephane Maarek 1:28 / 4:58

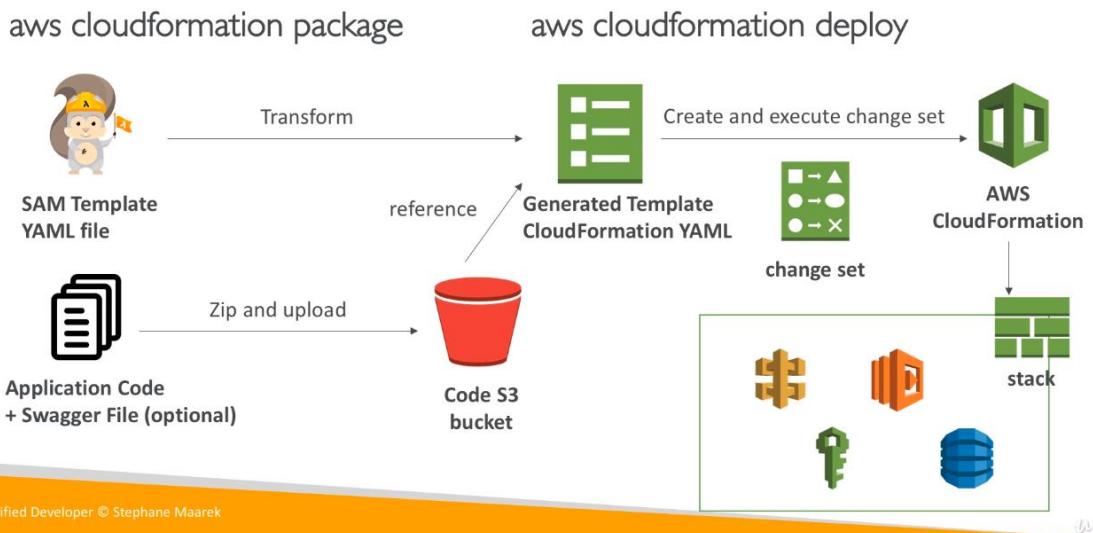
304. SAM Overview

AWS SAM – Recipe

- Transform Header indicates it's SAM template:
 - Transform: 'AWS::Serverless-2016-10-31'
- Write Code
 - AWS::Serverless::Function
 - AWS::Serverless::Api
 - AWS::Serverless::SimpleTable
- Package & Deploy:
 - aws cloudformation package / sam package
 - aws cloudformation deploy / sam deploy

AWS Certified Developer © Stephane Maarek 257 / 4:58

Deep dive into SAM deployment



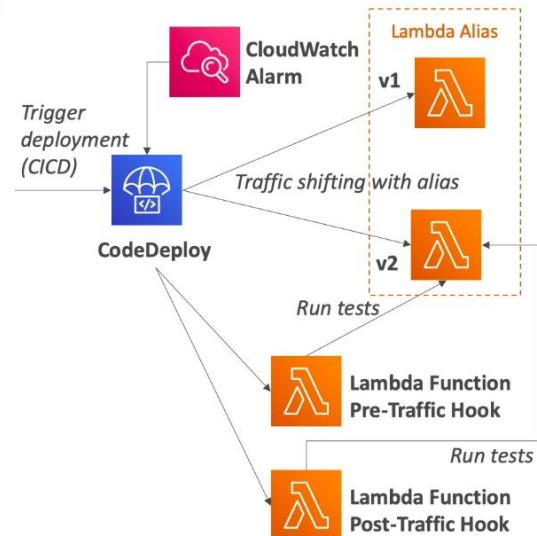
SAM Policy Templates

- List of templates to apply permissions to your Lambda Functions
- Full list available here: <https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-policy-templates.html#serverless-policy-template-table>
- Important examples:
 - **S3ReadPolicy**: Gives read only permissions to objects in S3
 - **SQSPollerPolicy**: Allows to poll an SQS queue
 - **DynamoDBCrudPolicy**: CRUD = create read update delete

```
MyFunction:
  Type: 'AWS::Serverless::Function'
  Properties:
    CodeUri: ${codeuri}
    Handler: hello.handler
    Runtime: python2.7
  Policies:
    - SQSPollerPolicy:
        QueueName:
          !GetAtt MyQueue.QueueName
```

SAM and CodeDeploy

- SAM framework natively uses CodeDeploy to update Lambda functions
- Traffic Shifting feature
- Pre and Post traffic hooks features to validate deployment (before the traffic shift starts and after it ends)
- Easy & automated rollback using CloudWatch Alarms



© Stephane Maarek

D3 session

Then V1 will be put away.



SAM – Exam Summary

- SAM is built on CloudFormation
- SAM requires the Transform and Resources sections
- Commands to know:
 - sam build: fetch dependencies and create local deployment artifacts
 - sam package: package and upload to Amazon S3, generate CF template
 - sam deploy: deploy to CloudFormation
- SAM Policy templates for easy IAM policy definition
- SAM is integrated with CodeDeploy to do deploy to Lambda aliases

© Stephane Maarek

D3 session

Amazon Cognito



- We want to give our users an identity so that they can interact with our application.
- **Cognito User Pools:**
 - Sign in functionality for app users
 - Integrate with API Gateway & Application Load Balancer
- **Cognito Identity Pools (Federated Identity):**
 - Provide AWS credentials to users so they can access AWS resources directly
 - Integrate with Cognito User Pools as an identity provider
- **Cognito Sync:**
 - Synchronize data from device to Cognito.
 - Is deprecated and replaced by AppSync
- **Cognito vs IAM:** "hundreds of users", "mobile users", "authenticate with SAML"

© Stephane Maarek

DD 2019

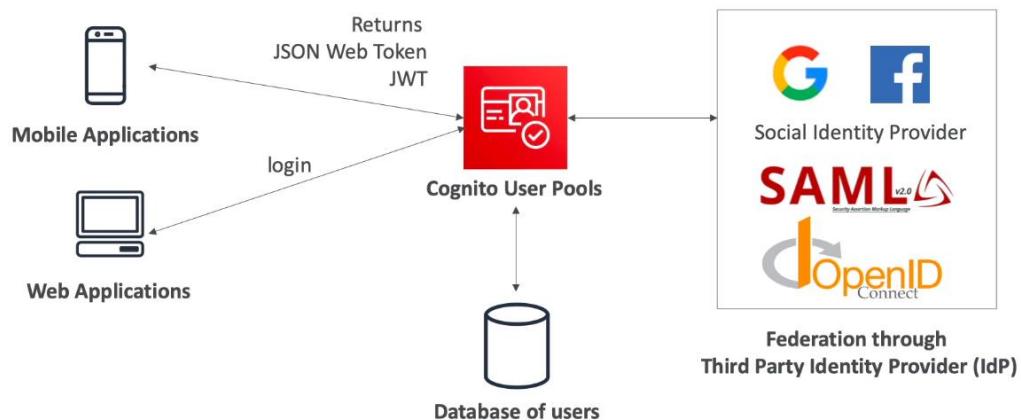
Cognito User Pools (CUP) – User Features

- Create a serverless database of user for your web & mobile apps
- Simple login: Username (or email) / password combination
- Password reset
- Email & Phone Number Verification
- Multi-factor authentication (MFA)
- Federated Identities: users from Facebook, Google, SAML...
- Feature: block users if their credentials are compromised elsewhere
- Login sends back a JSON Web Token (JWT)

© Stephane Maarek

DD 2019

Cognito User Pools (CUP) – Diagram



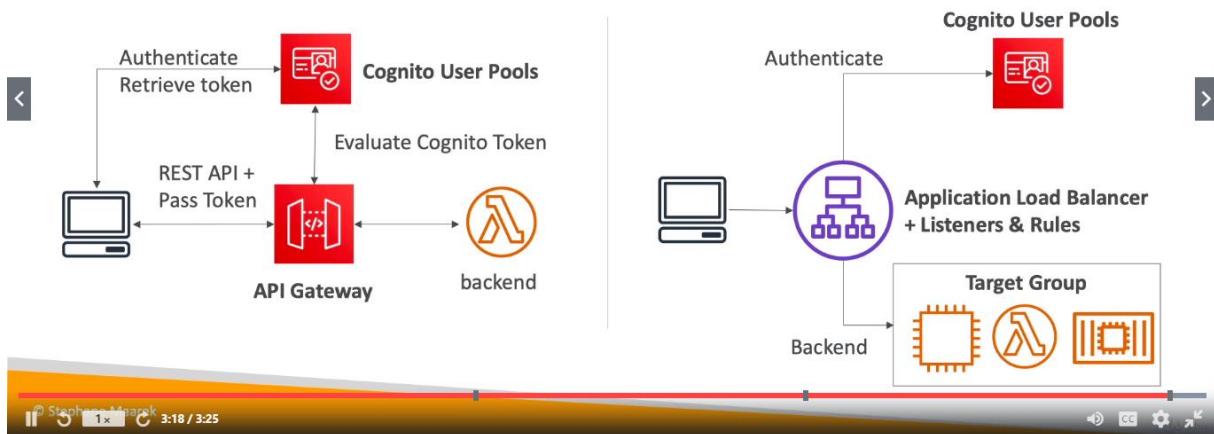
© Stephane Maarek

DD slides

315. Cognito User Pools

Cognito User Pools (CUP) - Integrations

- CUP integrates with API Gateway and Application Load Balancer



© Stephane Maarek

DD slides

3:18 / 3:25

Cognito User Pools – Lambda Triggers

- CUP can invoke a Lambda function synchronously on these triggers:

User Pool Flow	Operation	Description
Authentication Events	Pre Authentication Lambda Trigger	Custom validation to accept or deny the sign-in request
	Post Authentication Lambda Trigger	Event logging for custom analytics
	Pre Token Generation Lambda Trigger	Augment or suppress token claims
Sign-Up	Pre Sign-up Lambda Trigger	Custom validation to accept or deny the sign-up request
	Post Confirmation Lambda Trigger	Custom welcome messages or event logging for custom analytics
	Migrate User Lambda Trigger	Migrate a user from an existing user directory to user pools
Messages	Custom Message Lambda Trigger	Advanced customization and localization of messages
Token Creation	Pre Token Generation Lambda Trigger	Add or remove attributes in Id tokens

<https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-identity-pools-working-with-aws-lambda-triggers.html>

317. Cognito User Pools - Others

Cognito User Pools – Hosted Authentication UI

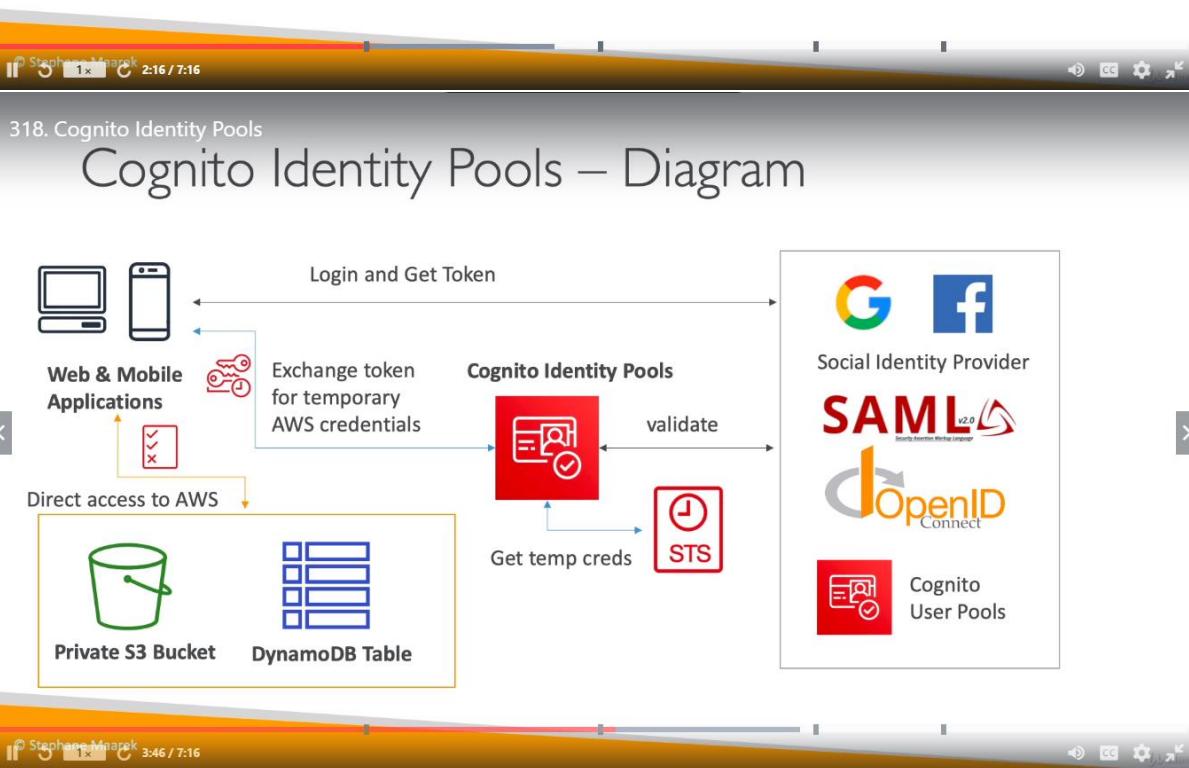
- Cognito has a hosted authentication UI that you can add to your app to handle sign-up and sign-in workflows
- Using the hosted UI, you have a foundation for integration with social logins, OIDC or SAML
- Can customize with a custom logo and custom CSS

<https://aws.amazon.com/blogs/aws/launch-amazon-cognito-user-pools-general-availability-app-integration-and-federation/>

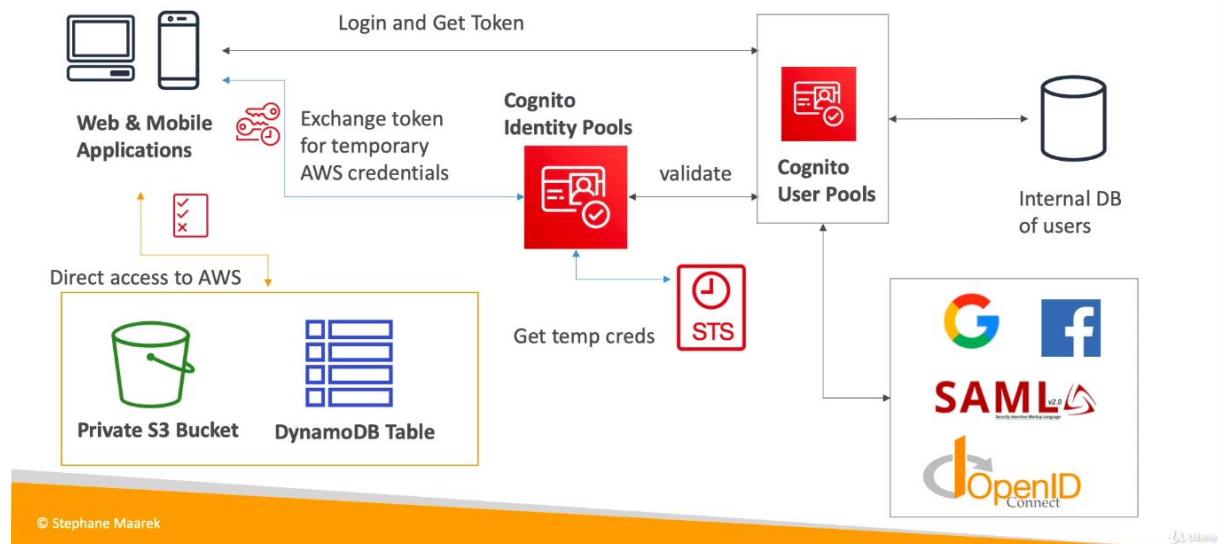
318. Cognito Identity Pools

Cognito Identity Pools (Federated Identities)

- Get identities for “users” so they obtain temporary AWS credentials
- Your identity pool (e.g identity source) can include:
 - Public Providers (Login with Amazon, Facebook, Google, Apple)
 - Users in an Amazon Cognito user pool
 - OpenID Connect Providers & SAML Identity Providers
 - Developer Authenticated Identities (custom login server)
 - Cognito Identity Pools allow for unauthenticated (guest) access
- Users can then access AWS services directly or through API Gateway
 - The IAM policies applied to the credentials are defined in Cognito
 - They can be customized based on the user_id for fine grained control



Cognito Identity Pools – Diagram with CUP



Cognito Identity Pools – IAM Roles

- Default IAM roles for authenticated and guest users
 - Define rules to choose the role for each user based on the user's ID
 - You can partition your users' access using policy variables
-
- IAM credentials are obtained by Cognito Identity Pools through STS
 - The roles must have a "trust" policy of Cognito Identity Pools

Cognito Identity Pools – Guest User example

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": [  
                "s3:GetObject"  
            ],  
            "Effect": "Allow",  
            "Resource": [  
                "arn:aws:s3:::mybucket/assets/my_picture.jpg"  
            ]  
        }  
    ]  
}
```

The screenshot shows a presentation slide with the following details:

- Title:** 318. Cognito Identity Pools
- Section:** Cognito Identity Pools – Policy variable on S3
- Code Block:** A JSON policy document is displayed, showing a single statement allowing S3GetObject on a specific file. The code is identical to the one shown in the previous slide.
- Annotations:** Two parts of the JSON code are highlighted with green boxes:
 - The condition part of the first statement: `Condition": {"StringLike": {"s3:prefix": ["${cognito-identity.amazonaws.com:sub}/*"]}}`
 - The resource part of the second statement: `"Resource": ["arn:aws:s3:::mybucket,${cognito-identity.amazonaws.com:sub}/*"]`
- Navigation:** The slide has navigation arrows on the left and right sides.
- Header:** The top header includes the author's name, Stephane Maarek, and the slide number, 318.
- Footer:** The bottom footer shows the presentation navigation controls and the slide number 645 / 716.

Cognito Identity Pools – DynamoDB

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:GetItem", "dynamodb:BatchGetItem", "dynamodb:Query",  
                "dynamodb:PutItem", "dynamodb:UpdateItem", "dynamodb:DeleteItem",  
                "dynamodb:BatchWriteItem"  
            ],  
            "Resource": [  
                "arn:aws:dynamodb:us-west-2:123456789012:table/MyTable"  
            ],  
            "Condition": {  
                "ForAllValues:StringEquals": {  
                    "dynamodb:LeadingKeys": [  
                        "${cognito-identity.amazonaws.com:sub}"  
                    ]  
                }  
            }  
        }  
    ]  
}
```

© Stephane Maarek

320. Cognito User Pools vs Cognito Identity Pools

Cognito User Pools vs Identity Pools



• Cognito User Pools:

- Database of users for your web and mobile application
- Allows to federate logins through Public Social, OIDC, SAML...
- Can customize the hosted UI for authentication (including the logo)]
- Has triggers with AWS Lambda during the authentication flow

• Cognito Identity Pools:

- Obtain AWS credentials for your users
- Users can login through Public Social, OIDC, SAML & Cognito User Pools
- Users can be unauthenticated (guests)
- Users are mapped to IAM roles & policies, can leverage policy variables

• CUP + CIP = manage user / password + access AWS services

321. Cognito Sync

Cognito Sync

- Deprecated – use AWS AppSync now
- Store preferences, configuration, state of app
- Cross device synchronization (any platform – iOS, Android, etc...)
- Offline capability (synchronization when back online)
- Store data in datasets (up to 1MB), up to 20 datasets to synchronize
- Push Sync: silently notify across all devices when identity data changes
- Cognito Stream: stream data from Cognito into Kinesis
- Cognito Events: execute Lambda functions in response to events



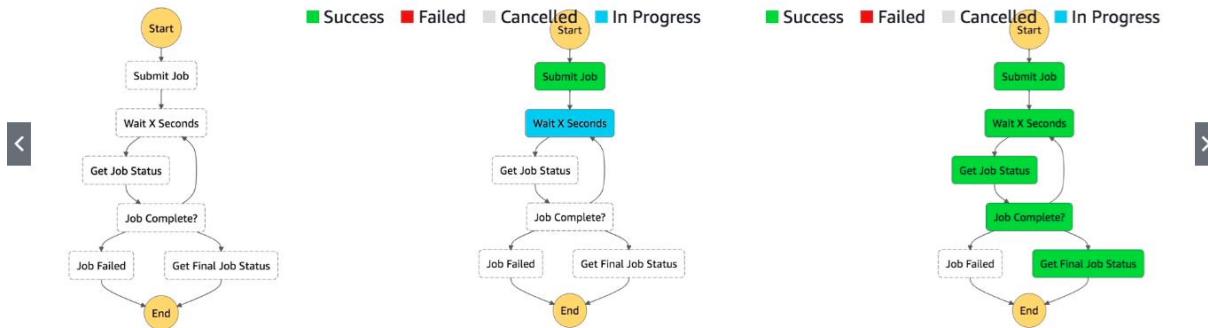
AWS Step Functions



- Build serverless visual workflow to orchestrate your Lambda functions
- Represent flow as a JSON state machine
- Features: sequence, parallel, conditions, timeouts, error handling...
- Can also integrate with EC2, ECS, On premise servers, API Gateway
- Maximum execution time of 1 year
- Possibility to implement human approval feature
- Use cases:
 - Order fulfillment
 - Data processing
 - Web applications
 - Any workflow



Visual workflow in Step Functions



Step Functions – Error Handling

- Any state can encounter runtime errors for various reasons:
 - State machine definition issues (for example, no matching rule in a Choice state)
 - Task failures (for example, an exception in a Lambda function)
 - Transient issues (for example, network partition events)
- By default, when a state reports an error, AWS Step Functions causes the execution to fail entirely.
- Retrying failures - **Retry**: IntervalSeconds, MaxAttempts, BackoffRate
- Moving on - **Catch**: ErrorEquals, Next
- Best practice is to include data in the error messages

Step Functions – Standard vs Express

	Standard Workflows	Express Workflows
Maximum duration	1 year.	5 minutes.
Supported execution start rate	Over 2,000 per second	Over 100,000 per second
Supported state transition rate	Over 4,000 per second per account	Nearly unlimited
Pricing	Priced per state transition. A state transition is counted each time a step in your execution is completed (more expensive)	Priced by the number of executions you run, their duration, and memory consumption (cheaper)
Execution history	Executions can be listed and described with Step Functions APIs, and visually debugged through the console. They can also be inspected in CloudWatch Logs by enabling logging on your state machine.	Executions can be inspected in CloudWatch Logs by enabling logging on your state machine.
Execution semantics	Exactly-once workflow execution.	At-least-once workflow execution.



AWS AppSync - Overview



- AppSync is a managed service that uses GraphQL
- GraphQL makes it easy for applications to get exactly the data they need.
- This includes combining data from **one or more sources**
 - NoSQL data stores, Relational databases, HTTP APIs...
 - Integrates with DynamoDB, Aurora, Elasticsearch & others
 - Custom sources with AWS Lambda
- Retrieve data in **real-time** with WebSocket or MQTT on WebSocket
- For mobile apps: local data access & data synchronization
- It all starts with uploading one **GraphQL schema**



324. AppSync Overview

GraphQL example

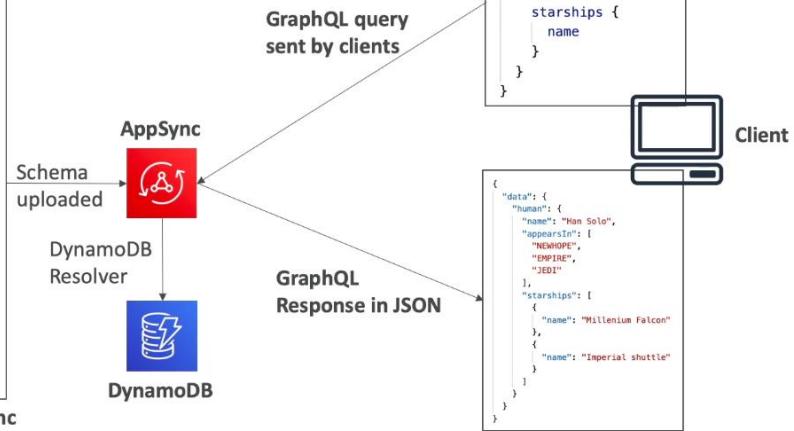
```
type Query {
    human(id: ID!): Human
}

type Human {
    name: String
    appearsIn: [Episode]
    starships: [Starship]
}

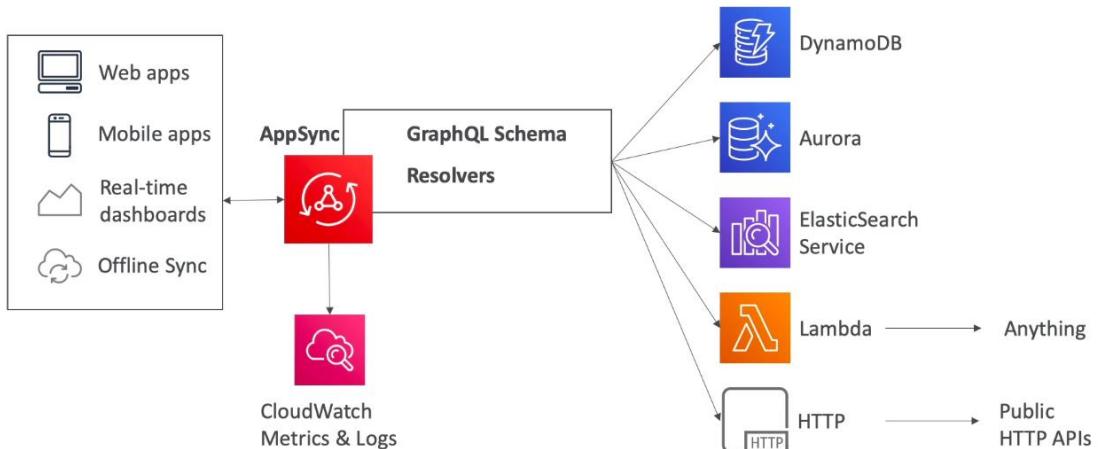
enum Episode {
    NEWHOPE
    EMPIRE
    JEDI
}

type Starship {
    name: String
}
```

GraphQL Schema on AppSync



AppSync Diagram



AppSync – Security

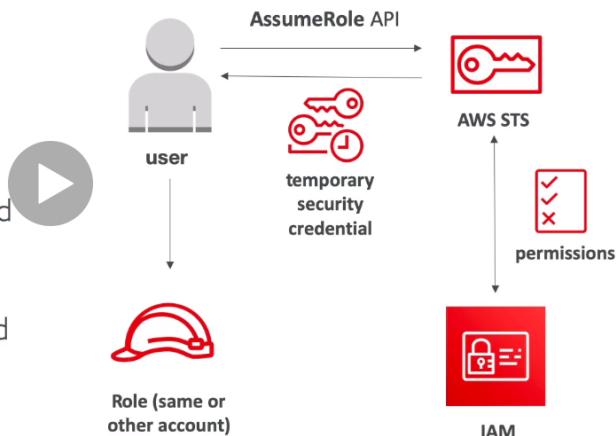
- There are four ways you can authorize applications to interact with your AWS AppSync GraphQL API:
 - API_KEY
 - AWS_IAM: IAM users / roles / cross-account access
 - OPENID_CONNECT: OpenID Connect provider / JSON Web Token
 - AMAZON_COGNITO_USER_POOLS
- For custom domain & HTTPS, use CloudFront in front of AppSync

The screenshot shows a presentation slide with the following details:

- Title:** 326. STS Overview
- Section:** AWS STS – Security Token Service
- Icon:** A red square icon containing a white clock symbol with the letters "STS" below it.
- Content:**
 - Allows to grant limited and temporary access to AWS resources (up to 1 hour).
 - AssumeRole:** Assume roles within your account or cross account
 - AssumeRoleWithSAML:** return credentials for users logged with SAML
 - AssumeRoleWithWebIdentity**
 - return creds for users logged with an IdP (Facebook Login, Google Login, OIDC compatible...)
 - AWS recommends against using this, and using Cognito Identity Pools instead
 - GetSessionToken:** for MFA, from a user or AWS account root user
 - GetFederationToken:** obtain temporary creds for a federated user
 - GetCallerIdentity:** return details about the IAM user or role used in the API call
 - DecodeAuthorizationMessage:** decode error message when an AWS API is denied

Using STS to Assume a Role

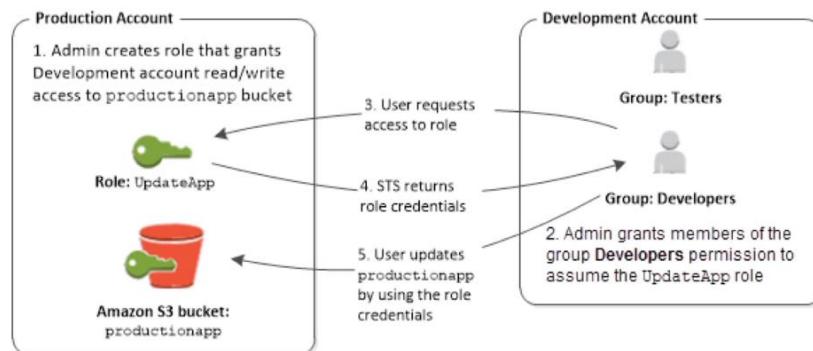
- Define an IAM Role within your account or cross-account
- Define which principals can access this IAM Role
- Use AWS STS (Security Token Service) to retrieve credentials and impersonate the IAM Role you have access to (AssumeRole API)
- Temporary credentials can be valid between 15 minutes to 1 hour



© Stephane Maarek

DD 30m

Cross account access with STS



https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_common-scenarios_aws-accounts.html

© Stephane Maarek

DD 30m

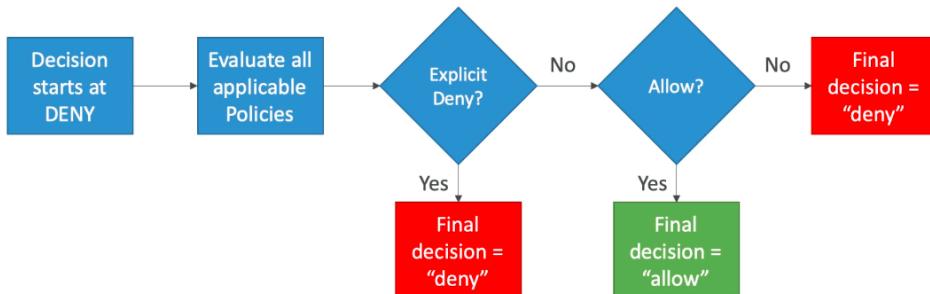
STS with MFA

- Use GetSessionToken from STS
- Appropriate IAM policy using IAM Conditions
- aws:MultiFactorAuthPresent:true
- Reminder; GetSessionToken returns:
 - Access ID
 - Secret Key
 - Session Token
 - Expiration date

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:StopInstances",
        "ec2:TerminateInstances"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "Bool": {
          "aws:MultiFactorAuthPresent": "true"
        }
      }
    }
  ]
}
```

Advanced IAM - Authorization Model Evaluation of Policies, simplified

1. If there's an explicit DENY, end decision and DENY
2. If there's an ALLOW, end decision with ALLOW
3. Else DENY



IAM Policies & S3 Bucket Policies

- IAM Policies are attached to users, roles, groups
- S3 Bucket Policies are attached to buckets
- When evaluating if an IAM Principal can perform an operation X on a bucket, the union of its assigned IAM Policies and S3 Bucket Policies will be evaluated.



AWS Certified Developer © Stephane Maarek

327. Advanced IAM

Example I

- IAM Role attached to EC2 instance, authorizes RW to "my_bucket"
- No S3 Bucket Policy attached
- => EC2 instance can read and write to "my_bucket"



Example 2

- IAM Role attached to EC2 instance, authorizes RW to “my_bucket”
- S3 Bucket Policy attached, explicit deny to the IAM Role
- => EC2 instance cannot read and write to “my_bucket”

AWS Certified Developer © Stephane Maarek

327. Advanced IAM

Example 3

- IAM Role attached to EC2 instance, no S3 bucket permissions
- S3 Bucket Policy attached, explicit RW allow to the IAM Role
- => EC2 instance can read and write to “my_bucket”

< >



Example 4

- IAM Role attached to EC2 instance, explicit deny S3 bucket permissions
- S3 Bucket Policy attached, explicit RW allow to the IAM Role
- => EC2 instance cannot read and write to "my_bucket"

Dynamic Policies with IAM

- How do you assign each user a /home/<user> folder in an S3 bucket?
- Option 1:
 - Create an IAM policy allowing georges to have access to /home/georges
 - Create an IAM policy allowing sarah to have access to /home/sarah
 - Create an IAM policy allowing matt to have access to /home/matt
 - ... One policy per user!
 - This doesn't scale
- Option 2:
 - Create one dynamic policy with IAM
 - Leverage the special policy variable \${aws:username}

Dynamic Policy example

```
{  
    "Sid": "AllowAllS3ActionsInUserFolder",  
    "Action": ["s3:*"],  
    "Effect": "Allow",  
    "Resource": ["arn:aws:s3:::my-company/home/${aws:username}/*"]  
}
```



Inline vs Managed Policies

- AWS Managed Policy
 - Maintained by AWS
 - Good for power users and administrators
 - Updated in case of new services / new APIs
- Customer Managed Policy
 - Best Practice, re-usable, can be applied to many principals
 - Version Controlled + rollback, central change management
- Inline
 - Strict one-to-one relationship between policy and principal
 - Policy is deleted if you delete the IAM principal



Granting a User Permissions to Pass a Role to an AWS Service

- To configure many AWS services, you must pass an IAM role to the service (this happens only once during setup)
- The service will later assume the role and perform actions
- Example of passing a role:
 - To an EC2 instance
 - To a Lambda function
 - To an ECS task
 - To CodePipeline to allow it to invoke other services
- For this, you need the IAM permission `iam:PassRole`
- It often comes with `iam:GetRole` to view the role being passed

© Stephane Maarek

DD 3000

IAM PassRole example

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "ec2:*"  
            ],  
            "Resource": "*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": "iam:PassRole",  
            "Resource": "arn:aws:iam::123456789012:role/S3Access"  
        }  
    ]  
}
```

© Stephane Maarek

DD 3000

Can a role be passed to any service?

- No: Roles can only be passed to what their trust allows
- A *trust policy* for the role that allows the service to assume the role

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "TrustPolicyStatementThatAllowsEC2ServiceToAssumeTheAttachedRole",  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "ec2.amazonaws.com"  
            },  
            "Action": "sts:AssumeRole"  
        }  
    ]  
}
```

329. Directory Services - Overview

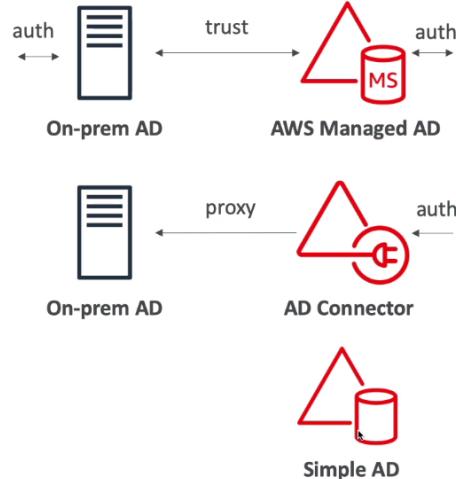
What is Microsoft Active Directory (AD)?

- Found on any Windows Server with AD Domain Services
- Database of objects: User Accounts, Computers, Printers, File Shares, Security Groups
- Centralized security management, create account, assign permissions
- Objects are organized in trees
- A group of trees is a forest

The diagram illustrates the structure of Active Directory. At the top, there is a computer monitor icon labeled "Domain Controller". Below it, the text "John Password" is shown. Five smaller computer monitor icons are arranged horizontally below the domain controller, each connected by a line to the "John Password" text, representing client computers or workstations.

AWS Directory Services

- AWS Managed Microsoft AD
 - Create your own AD in AWS, manage users locally, supports MFA
 - Establish “trust” connections with your on-premise AD
- AD Connector
 - Directory Gateway (proxy) to redirect to on-premise AD
 - Users are managed on the on-premise AD
- Simple AD
 - AD-compatible managed directory on AWS
 - Cannot be joined with on-premise AD



Why encryption?

Encryption in flight (SSL)

- Data is encrypted before sending and decrypted after receiving
- SSL certificates help with encryption (HTTPS)
- Encryption in flight ensures no MITM (man in the middle attack) can happen



Why encryption?

Server side encryption at rest

- Data is encrypted after being received by the server
- Data is decrypted before being sent
- It is stored in an encrypted form thanks to a key (usually a data key)
- The encryption / decryption keys must be managed somewhere and the server must have access to it



AWS Certified Developer © Stephane Maarek

33/34

Why encryption?

Client side encryption

- Data is encrypted by the client and never decrypted by the server
- Data will be decrypted by a receiving client
- The server should not be able to decrypt the data
- Could leverage Envelope Encryption



AWS Certified Developer © Stephane Maarek

34/34

AWS KMS (Key Management Service)



- Anytime you hear “encryption” for an AWS service, it’s most likely KMS
- Easy way to control access to your data, AWS manages keys for us
- Fully integrated with IAM for authorization
- Seamlessly integrated into:
 - Amazon EBS: encrypt volumes
 - Amazon S3: Server side encryption of objects
 - Amazon Redshift: encryption of data
 - Amazon RDS: encryption of data
 - Amazon SSM: Parameter store
 - Etc...
- But you can also use the CLI / SDK

The screenshot shows a presentation slide with the following details:

- Title:** 332. KMS Overview
- Section:** KMS – Customer Master Key (CMK) Types
- Content:**
 - Symmetric (AES-256 keys)
 - First offering of KMS, single encryption key that is used to Encrypt and Decrypt
 - AWS services that are integrated with KMS use Symmetric CMKs
 - Necessary for envelope encryption
 - You never get access to the Key unencrypted (must call KMS API to use)
 - Asymmetric (RSA & ECC key pairs)
 - Public (Encrypt) and Private Key (Decrypt) pair
 - Used for Encrypt/Decrypt, or Sign/Verify operations
 - The public key is downloadable, but you can't access the Private Key unencrypted
 - Use case: encryption outside of AWS by users who can't call the KMS API
- Navigation:** Left and right arrows for navigating through the presentation.
- Player controls:** Standard video player controls (play/pause, volume, etc.) are visible at the bottom.

AWS KMS (Key Management Service)

- Able to fully manage the keys & policies:
 - Create
 - Rotation policies
 - Disable
 - Enable
- Able to audit key usage (using CloudTrail)
- Three types of Customer Master Keys (CMK):
 - AWS Managed Service Default CMK: free
 - User Keys created in KMS: \$1 / month
 - User Keys imported (must be 256-bit symmetric key): \$1 / month
- + pay for API call to KMS (\$0.03 / 10000 calls)



AWS KMS 101

- Anytime you need to share sensitive information... use KMS
 - Database passwords
 - Credentials to external service
 - Private Key of SSL certificates
- The value in KMS is that the CMK used to encrypt data can never be retrieved by the user, and the CMK can be rotated for extra security



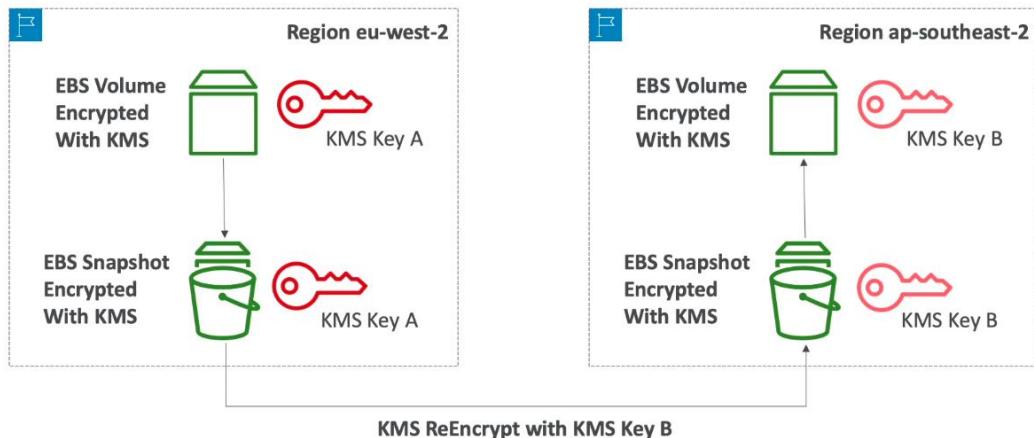
AWS KMS 101

- Never ever store your secrets in plaintext, especially in your code!
 - Encrypted secrets can be stored in the code / environment variables
 - KMS can only help in encrypting up to 4KB of data per call
 - If data > 4 KB, use envelope encryption
-
- To give access to KMS to someone:
 - Make sure the Key Policy allows the user
 - Make sure the IAM Policy allows the API calls

© Stephane Maarek

DD 10min

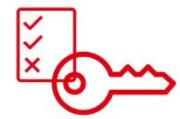
Copying Snapshots across regions



© Stephane Maarek

DD 10min

KMS Key Policies



- Control access to KMS keys, “similar” to S3 bucket policies
- Difference: you cannot control access without them
- **Default KMS Key Policy:**
 - Created if you don't provide a specific KMS Key Policy
 - Complete access to the key to the root user = entire AWS account
 - Gives access to the IAM policies to the KMS key
- **Custom KMS Key Policy:**
 - Define users, roles that can access the KMS key
 - Define who can administer the key
 - Useful for cross-account access of your KMS key

The screenshot shows a presentation slide with the following details:

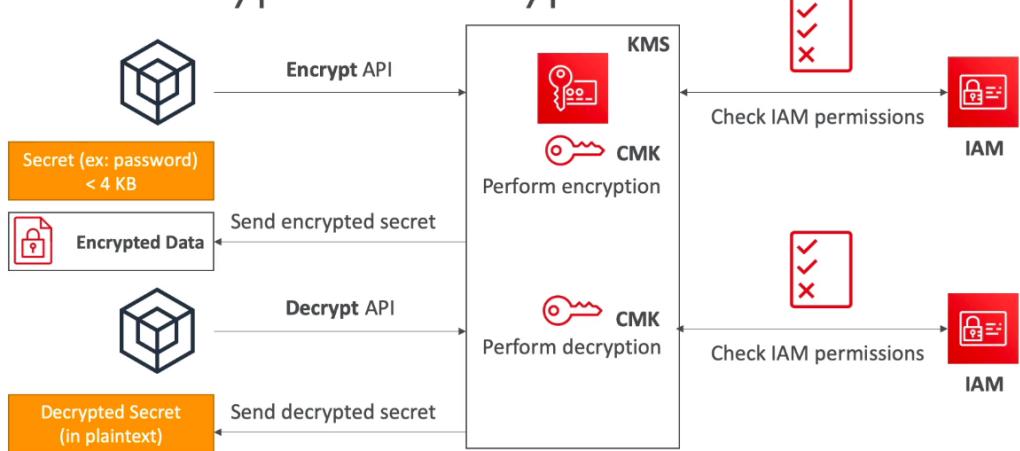
- Title:** 332. KMS Overview
- Section:** Copying Snapshots across accounts
- Content:** A numbered list of steps:
 1. Create a Snapshot, encrypted with your own CMK
 2. Attach a KMS Key Policy to authorize cross-account access
 3. Share the encrypted snapshot
 4. (in target) Create a copy of the Snapshot, encrypt it with a KMS Key in your account
 5. Create a volume from the snapshot
- Code Block:** A box containing a JSON-based KMS Key Policy document.

```
{  
    "Sid": "Allow use of the key with destination account",  
    "Effect": "Allow",  
    "Principal": {  
        "AWS": "arn:aws:iam::TARGET-ACCOUNT-ID:role/ROLENAMESPACE"  
    },  
    "Action": [  
        "kms:Decrypt",  
        "kms>CreateGrant"  
    ],  
    "Resource": "*",  
    "Condition": {  
        "StringEquals": {  
            "kms:ViaService": "ec2.REGION.amazonaws.com",  
            "kms:CallerAccount": "TARGET-ACCOUNT-ID"  
        }  
    }  
}
```

KMS Key Policy
- Navigation:** Left and right arrows for navigating through the slide.
- Bottom Bar:** Includes icons for search, refresh, and other presentation controls.

How does KMS work?

API – Encrypt and Decrypt



© Stephane Maarek

334

334. KMS Encryption Patterns and Envelope Encryption

Envelope Encryption

- KMS Encrypt API call has a limit of 4 KB
- If you want to encrypt >4 KB, we need to use Envelope Encryption
- The main API that will help us is the GenerateDataKey API

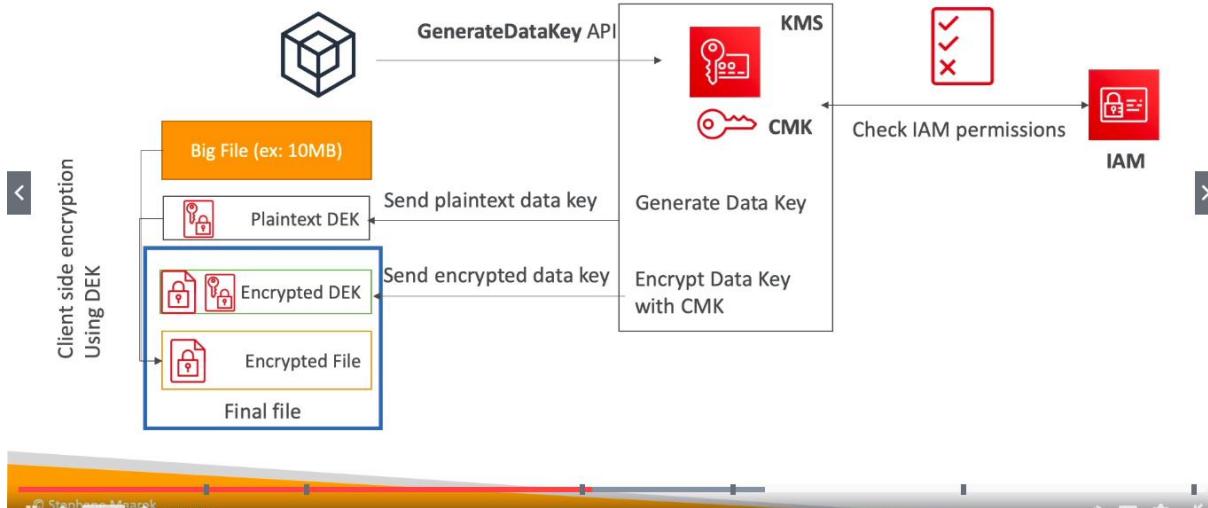
- For the exam: anything over 4 KB of data that needs to be encrypted must use the Envelope Encryption == GenerateDataKey API

© Stephane Maarek 158 / 726

334

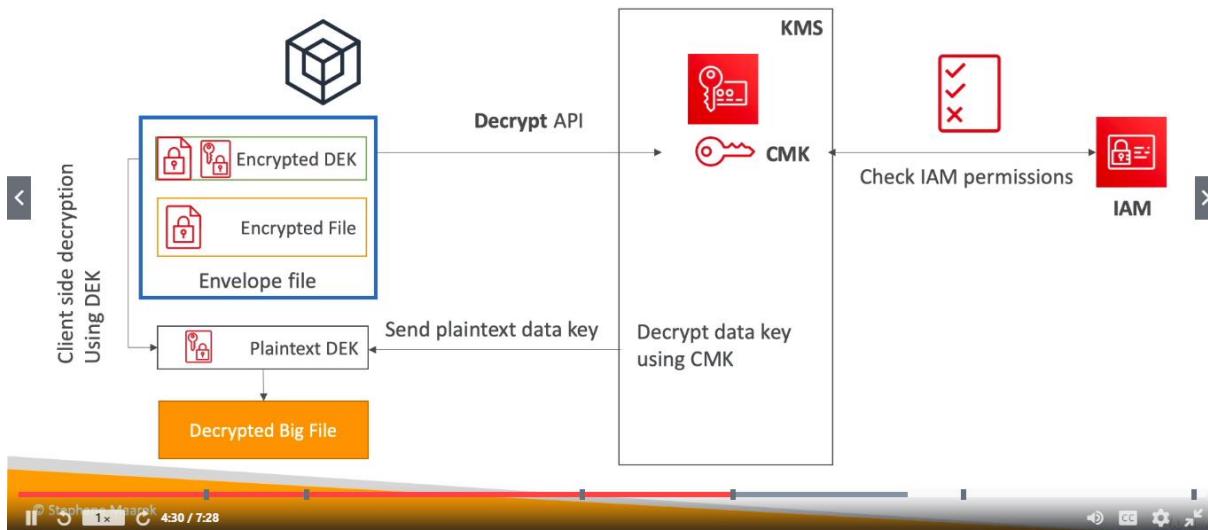
334. KMS Encryption Patterns and Envelope Encryption

GenerateDataKey API



334. KMS Encryption Patterns and Envelope Encryption

Decrypt envelope data



Encryption SDK



- The AWS Encryption SDK implemented Envelope Encryption for us
- The Encryption SDK also exists as a CLI tool we can install
- Implementations for Java, Python, C, JavaScript

< • Feature - Data Key Caching:

- re-use data keys instead of creating new ones for each encryption
- Helps with reducing the number of calls to KMS with a security trade-off
- Use LocalCryptoMaterialsCache (max age, max bytes, max number of messages)

KMS Symmetric – API Summary



- Encrypt: encrypt up to 4 KB of data through KMS
- GenerateDataKey: generates a unique symmetric data key (DEK)
 - returns a plaintext copy of the data key
 - AND a copy that is encrypted under the CMK that you specify
- GenerateDataKeyWithoutPlaintext:
 - Generate a DEK to use at some point (not immediately)
 - DEK that is encrypted under the CMK that you specify (must use Decrypt later)
- Decrypt: decrypt up to 4 KB of data (including Data Encryption Keys)
- GenerateRandom: Returns a random byte string

KMS Request Quotas

- When you exceed a request quota, you get a ThrottlingException:

```
You have exceeded the rate at which you may call KMS. Reduce the frequency of your calls.  
(Service: AWSKMS; Status Code: 400; Error Code: ThrottlingException; Request ID: <ID>)
```

- To respond, use exponential backoff (backoff and retry)
- For cryptographic operations, they share a quota
- This includes requests made by AWS on your behalf (ex: SSE-KMS)
- For GenerateDataKey, consider using DEK caching from the Encryption SDK
- You can request a Request Quotas increase through API or AWS support

The screenshot shows a presentation slide with the following details:

- Header:** © Stephane Maarek, CC BY-SA
- Title:** 336. KMS Limits
- Section:** KMS Request Quotas
- Table:** API operation vs. Request quotas (per second)
- Table Data:**

API operation	Request quotas (per second)
Decrypt Encrypt GenerateDataKey (symmetric) GenerateDataKeyWithoutPlaintext (symmetric) GenerateRandom ReEncrypt Sign (asymmetric) Verify (asymmetric)	These shared quotas vary with the AWS Region and the type of CMK used in the request. Each quota is calculated separately. Symmetric CMK quota: <ul style="list-style-type: none">5,500 (shared)10,000 (shared) in the following Regions:<ul style="list-style-type: none">us-east-2, ap-southeast-1, ap-southeast-2, ap-northeast-1, eu-central-1, eu-west-230,000 (shared) in the following Regions:<ul style="list-style-type: none">us-east-1, us-west-2, eu-west-1 Asymmetric CMK quota: <ul style="list-style-type: none">500 (shared) for RSA CMKs300 (shared) for Elliptic curve (ECC) CMKs
- Navigation:** < >
- Footer:** © Stephane Maarek, 227 / 249, CC BY-SA

S3 Encryption for Objects

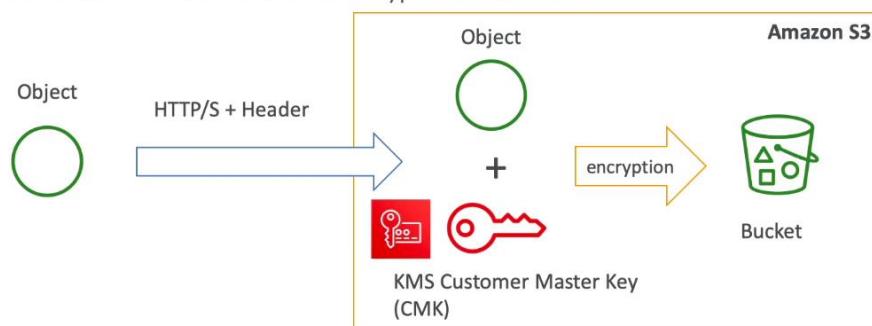
- There are 4 methods of encrypting objects in S3
 - SSE-S3: encrypts S3 objects using keys handled & managed by AWS
 - SSE-KMS: leverage AWS Key Management Service to manage encryption keys
 - SSE-C: when you want to manage *your own* encryption keys
 - Client Side Encryption
- It's important to understand which ones are adapted to which situation for the exam

© Stephane Maarek

D3.3000

SSE-KMS

- SSE-KMS: encryption using keys handled & managed by KMS
- KMS Advantages: user control + audit trail
- Object is encrypted server side
- Must set header: "x-amz-server-side-encryption": "aws:kms"



© Stephane Maarek

D3.3000

SSE-KMS Deep Dive

- SSE-KMS leverages the GenerateDataKey & Decrypt KMS API calls
- These KMS API calls will show up in CloudTrail, helpful for logging
- To perform SSE-KMS, you need:
 - A KMS Key Policy that authorizes the user / role
 - An IAM policy that authorizes access to KMS
 - Otherwise you will get an access denied error
- S3 calls to KMS for SSE-KMS count against your KMS limits
 - If throttling, try exponential backoff
 - If throttling, you can request an increase in KMS limits
 - The service throttling is KMS, not Amazon S3



S3 Bucket Policies – Force SSL

```
{
  "Id": "ExamplePolicy",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSSLRequestsOnly",
      "Action": "s3:*",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:s3:::awsexamplebucket",
        "arn:aws:s3:::awsexamplebucket/*"
      ],
      "Condition": {
        "Bool": {
          "aws:SecureTransport": "false"
        }
      },
      "Principal": "*"
    }
  ]
}
```

- To force SSL, create an S3 bucket policy with a **DENY** on the condition `aws:SecureTransport = false`
- Note: Using an allow on `aws:SecureTransport = true` would allow anonymous `GetObject` if using SSL
- Read more here: <https://aws.amazon.com/premiumsupport/knowledge-center/s3-bucket-policy-for-config-rule/>



S3 Bucket Policy – Force Encryption of SSE-KMS

1. Deny incorrect encryption header: make sure it includes aws:kms (== SSE-KMS)
 2. Deny no encryption header to ensure objects are not uploaded un-encrypted
- Note: could swap 2) for S3 default encryption of SSE-KMS

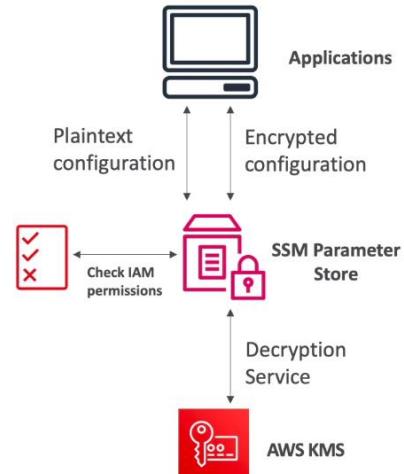
```
{  
    "Version": "2012-10-17",  
    "Id": "PutObjPolicy",  
    "Statement": [  
        {  
            "Sid": "DenyIncorrectEncryptionHeader",  
            "Effect": "Deny",  
            "Principal": "*",  
            "Action": "s3:PutObject",  
            "Resource": "arn:aws:s3:::<bucket_name>/*",  
            "Condition": {  
                "StringNotEquals": {  
                    "s3:x-amz-server-side-encryption": "aws:kms"  
                }  
            }  
        },  
        {  
            "Sid": "DenyUnEncryptedObjectUploads",  
            "Effect": "Deny",  
            "Principal": "*",  
            "Action": "s3:PutObject",  
            "Resource": "arn:aws:s3:::<bucket_name>/*",  
            "Condition": {  
                "Null": {  
                    "s3:x-amz-server-side-encryption": true  
                }  
            }  
        }  
    ]  
}
```

© Stephane Maarek

↓.s3naw

SSM Parameter Store

- Secure storage for configuration and secrets
- Optional Seamless Encryption using KMS
- Serverless, scalable, durable, easy SDK
- Version tracking of configurations / secrets
- Configuration management using path & IAM
- Notifications with CloudWatch Events
- Integration with CloudFormation

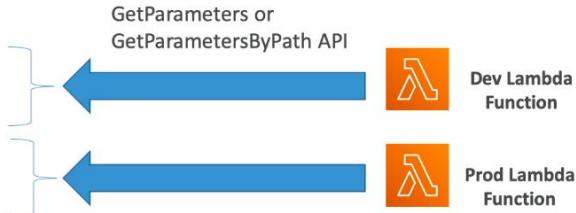


© Stephane Maarek

↓.s3naw

SSM Parameter Store Hierarchy

- /my-department/
 - my-app/
 - dev/
 - db-url
 - db-password
 - prod/
 - db-url
 - db-password
 - other-app/
- /other-department/
- /aws/reference/secretsmanager/secret_ID_in_Secrets_Manager
- /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2



© Stephane Maarek

DJ Schow

Standard and advanced parameter tiers

	Standard	Advanced
Total number of parameters allowed (per AWS account and Region)	10,000	100,000
Maximum size of a parameter value	4 KB	8 KB
Parameter policies available	No	Yes
Cost	No additional charge	Charges apply
Storage Pricing	Free	\$0.05 per advanced parameter per month
API Interaction Pricing (higher throughput = up to 1000 Transactions per second)	Standard Throughput: free Higher Throughput: \$0.05 per 10,000 API interactions	Standard Throughput: \$0.05 per 10,000 API interactions Higher Throughput: \$0.05 per 10,000 API interactions

© Stephane Maarek

DJ Schow

Parameters Policies (for advanced parameters)

- Allow to assign a TTL to a parameter (expiration date) to force updating or deleting sensitive data such as passwords
- Can assign multiple policies at a time

Expiration (to delete a parameter)

```
{  
  "Type": "Expiration",  
  "Version": "1.0",  
  "Attributes": {  
    "Timestamp": "2020-12-02T21:34:33.000Z"  
  }  
}
```

ExpirationNotification (CW Events)

```
{  
  "Type": "ExpirationNotification",  
  "Version": "1.0",  
  "Attributes": {  
    "Before": "15",  
    "Unit": "Days"  
  }  
}
```

NoChangeNotification (CW Events)

```
{  
  "Type": "NoChangeNotification",  
  "Version": "1.0",  
  "Attributes": {  
    "After": "20",  
    "Unit": "Days"  
  }  
}
```

The screenshot shows a slide titled '342. Secrets Manager - Overview' with the heading 'AWS Secrets Manager'. It features a red icon of a padlock inside a shield. Below the heading is a bulleted list of features:

- Newer service, meant for storing secrets
- Capability to force rotation of secrets every X days
- Automate generation of secrets on rotation (uses Lambda)
- Integration with Amazon RDS (MySQL, PostgreSQL, Aurora)
- Secrets are encrypted using KMS
- Mostly meant for RDS integration



SSM Parameter Store vs Secrets Manager

- **Secrets Manager (\$\$\$):**
 - Automatic rotation of secrets with AWS Lambda
 - Integration with RDS, Redshift, DocumentDB
 - KMS encryption is mandatory
 - Can integration with CloudFormation
- **SSM Parameter Store (\$):**
 - Simple API
 - No secret rotation
 - KMS encryption is optional
 - Can integration with CloudFormation
 - Can pull a Secrets Manager secret using the SSM Parameter Store API

CloudWatch Logs - Encryption



- You can encrypt CloudWatch logs with KMS keys
 - Encryption is enabled at the log group level, by associating a CMK with a log group, either when you create the log group or after it exists.
 - You cannot associate a CMK with a log group using the CloudWatch console.
-
- You must use the CloudWatch Logs API:
 - `associate-kms-key` : if the log group already exists
 - `create-log-group`: if the log group doesn't exist yet

CodeBuild Security



- To access resources in your VPC, make sure you specify a VPC configuration for your CodeBuild
- Secrets in CodeBuild:
 - Don't store them as plaintext in environment variables
 - Instead...
 - Environment variables can reference parameter store parameters
 - Environment variables can reference secrets manager secrets



AWS SES – Simple Email Service

- Send emails to people using:
 - SMTP interface
 - Or AWS SDK
- Ability to receive email. Integrates with:
 - S3
 - SNS
 - Lambda
- Integrated with IAM for allowing to send emails

AWS Databases Summary

- RDS: Relational databases, OLTP
 - PostgreSQL, MySQL, Oracle...
 - Aurora + Aurora Serverless
 - Provisioned database
- DynamoDB: NoSQL DB
 - Managed, Key Value, Document
 - Serverless
- ElastiCache: In memory DB
 - Redis / Memcached
 - Cache capability
- Redshift: OLAP – Analytic Processing
 - Data Warehousing / Data Lake
 - Analytics queries
- Neptune: Graph Database
- DMS: Database Migration Service
- DocumentDB: managed MongoDB for AWS

© Stephane Maarek

350. Amazon Certificate Manager (ACM)

AWS Certificate Manager (ACM)

- To host public SSL certificates in AWS, you can:
 - Buy your own and upload them using the CLI
 - Have ACM provision and renew public SSL certificates for you (free of cost)
- ACM loads SSL certificates on the following integrations:
 - Load Balancers (including the ones created by EB)
 - CloudFront distributions
 - APIs on API Gateways
- SSL certificates is overall a pain to manually manage, to ACM is great to leverage in your AWS infrastructure!

```
graph TD; User[Users] -- "Public www HTTPS Request" --> ALB[ALB  
SSL Termination]; ALB <--> ACM[ACM]; ALB -- "Private AWS HTTP Request" --> EC2[EC2];
```

Less CPU cost in EC2
Thanks to SSL termination for the ELB

© Stephane Maarek

State of learning checkpoint

- Let's look how far we've gone on our learning journey
- <https://aws.amazon.com/certification/certified-developer-associate/>