Coding Conventions and Guidelines

Naming Conventions

General guidelines

Guideline	Name	Status	Description
GG-01	Succinct Naming	Proposed /Accepted 03.11.2021	Long names can make code hard to read and cumbersome to write. We should strive to chose the shortest name possible without hiding the meaning behind it. Avoid abbreviations: ▪ People might not know what the abbreviation stands for ▪ People might introduce new (wrong) abbreviations ▪ Our team is just starting with AWS and CDK, so having the names spelled out makes it easier for us to understand the meaning. Exceptions where abbreviations might be acceptable are: ▪ **db** for Database ▪ **id** for Identifier
GG-02	Clear Naming	Proposed /Accepted 03.11.2021	Try to avoid ambiguous names. `policy` might be a good name if there is only one policy in the current context, but as soon as there are multiple policies the name looses it's meaning. Try to use concrete names as `accessDbPolicy` or `s3ReadAccessPolicy`.
GG-03	Exact Naming / "Names" should not lie	Proposed /Accepted 03.11.2021	Lying in this context means that the name implies one thing while the actual object behind the name is another thing. A very basic example might be a variable `int user`. One can assume that in this context the variable holds the userId and not the user itself. You might be able to deduct the meaning from the existing context. But even better is if the variable name does not lie. In this case `int userId` would be the more appropriate name. "Lying" names can easily be introduced during development or refactoring, when there are many moving parts. Always make sure to review your changes and verify that the meaning of the names is correct.
GG-04	Consider context	Proposed /Accepted 03.11.2021	It is important to use exact naming, but in some cases the context gives us the information we need. Consider we want to create a stack that deploys two lambda functions (readFn and writeFn) with custom execution roles. If we define both roles and lambdas in the stack itself then we should include to information on what lambda or role we are talking about in the variable name. We have the functions `read-function` and `write-function`, as well as the execution roles `read-function-execution-role` and `write-function-exeuction-role`. Now if we create a construct that will create the function and execution role, we can call the construct once for each function we want to create. The id of those constructs could be `read-handler` and `write-handler`. In the construct itself we can id the function simply as `function` and the role as `execution-role`. In the context of our CDK project, the ids would be `read-handler.function`, `read-handler.execution-role`, `write-handler.function` and `write-handler.execution-role`.
GG-05	No external variables beyond the app	Proposed /Accepted 03.11.2021	Configuration coming from environment variables must always be resolved in the app on the top level. If a stack or construct requires one of those values, it should be introduced as a field in the stacks or constructs props object.
GG-06	Describe Inputs /Outputs in readme	Proposed /Accepted 03.11.2021	
GG-07	Avoid many parameters	Proposed /Accepted 08.12.2021	Methods (and constructors) should only have a few parameters. Consider introducing a Props/Options class (using @Data and @Builder lombok annotations) to pass parameters to a method. This is especially important if you have methods with lots of parameters of the same type, as it is very easy to introduce regressions in a code base when such methods are modified.
GG-08	Validate external input	Proposed /Accepted 08.12.2021	We need to validate all external input. External input is everything that comes from outside of our application. This can be data from a database or the request parameters which whom the lambda or fargate applications have been called.
GG-09	Separate Business Logic from Data Access and Controllers /Handlers	Proposed /Accepted 08.12.2021	We should have a clear separations of concerns. The business logic layer should not care about data access, nor about request parameters, which makes it easier for testing.

GG-10	Boy/Girl-scouts mentality	Proposed /Accepted 08.12.2021	Use the Boy/Girl-scouts mentality of leaving a place in a better state than you found it. If you touch code and you notice some issues that can be easily fixed (missing comments, wrong formatting, incorrect variable names), then try to fix them. Make sure that your changes are covered by unit tests before doing the changes. The tests should be written upfront.
GG-11	Resilience	Proposed /Accepted 08.12.2021	With serverless architecture we have to expect that a service might not respond directly or return an error on the first call. We need to write code that takes this into account. Make sure to employ strategies like retries and exponential backoff to recover from failures if possible. A db connection in a lambda might time out. If that's the case we can recreate the connection and retry. Make sure to take responses of other services into account. Retries should not be used in each case, only when it's feasable (Resource timed out, rate limit hit, db connection expired).
GG-12	KISS > DRY	Proposed /Accepted 08.12.2021	Keeping your code simple is in microservices usually better then making sure you don't repeat yourself since the latter can lead to more complex code especially when resorting to generics and similar language features.
GG-13	(S)OLID	Proposed /Accepted 08.12.2021	Single responsibility, Does this class do only one thing, Does it's name explain what that purpose is clearly, do the public methods explain what functionality it offers
GG-14	Package by feature	Proposed /Accepted 08.12.2021	Increases code cohesion and decreases dependencies between pieces of code.

IDs of constructs

Guideline	Name	Status	Description
CID-01	kebap-case	Proposed /Accepted 03.11.2021	The IDs of constructs must use kebap case (all lower case letters, words separated by `-`). **Reasoning:** CDK itself uses CamelCase for naming. We could have opted also to use CamelCase for consistency. The reason we chose kebap-case is that it clearly distinguishes on which constructs have been created by us and which have been created as part of a high level construct. Examples: ▪ execution-role ▪ s3-read-policy
CID-02	function or construct suffix	Proposed /Accepted 03.11.2021	The id should include the name of the construct if possible. A policy should end in `-policy`, a role in `-role` etc. This is so that we avoid naming conflicts. In some cases it might be possible to leave the construct away, if the meaning is obvious.

Names of constructs

Guideline	Name	Status	Description
CN-01	kebap-case	Proposed /Accepted 03.11.2021	The names of constructs must use kebap case (all lower case letters, words separated by `-`). **Important:** Some resources do not allow using special characters like `-`. For those we should strip the invalid characters from the name. Examples: • execution-role • s3-read-policy
CN-02	function or construct suffix	Proposed /Accepted 03.11.2021	The name should include the name of the construct if possible. A policy should end in `-policy`, a role in `-role` etc. This is so that we avoid naming conflicts. In some cases it might be possible to leave the construct away, if the meaning is obvious.

CN-03	Resource prefix for global constructs	Proposed /Accepted 03.11.2021	Names of certain AWS services or objects need to be unique within the account or even globally. For such constructs we must always add the resource prefix, so we avoid conflicts when the stack is deployed multiple times in the same account. For example an S3 bucket name needs to be unique globally. So for bucket names we always have to include the prefix.
CN-04	No resource prefix for local constructs	Proposed /Accepted 03.11.2021	Names for local objects do not need to be prefixed. Local objects are usually part of some service (ex: User Groups are local objects to CognitoUserPool). There can be 200 different user pools, all with a group named "Admin". As the groups belong to the user pool, the "Admin" group from pool A will not conflict with the "Admin" group from pool B.
CN-05	Hyphen at end of prefix	Proposed /Accepted 03.11.2021	Adding a hyphen at the end of your prefix makes the construct names a bit nicer to read. Example: ▪ **Bad example:** With prefix `zelenay` and resources `read-handler` and `rds`

Working with GIT

Guideline	Name	Status	Description
GIT-01	branch names	Proposed /Accepted 03.11.2021	Branch names should follow the naming format `<FUNCTION_OF_BRANCH>/<JIRA_ID>-<SHORT_DESCRIPTION>`. `<FUNTION_OF_BRANCH>` can be `feature` for a new feature that is being developed or `fix` for hotfix branches. `<JIRA_ID>` is the ticket number of the ticket being worked on in this branch. `<SHORT_DESCRIPTION>` is a short description on what the branch is about. This can be the same as the Jira ticket summary or it can be something different (for example if you only tackle part of a ticket) The main/master branch is exempt from this guideline. Examples: ▪ `feature/CASS-8642-integrating-lb-with-apigw` ▪ `fix/CASS-8642-waf-association-is-creating-before-api-stage-is-ready`
GIT-02	commit messages	Proposed /Accepted 03.11.2021	Commit messages should follow the format `<KIND_OF_CHANGE>(<JIRA_ID>):  <SUBJECT>[<BODY>]` `<KIND_OF_CHANGE>` refers to one of the multiple existing key words used for conventional commits. We use following keywords: ▪ BREAKING_CHANGE: (Major Version) - Change introduces breaking changes. ▪ FEAT: (Minor Version) - Change introduces a new feature without breaking existing functionality. ▪ FIX: (Patch Version) - Change fixes an existing bug or behaviour without introducing new functionality. ▪ CHORE: (Patch Version) - Is used for basic work that does not relate to program logic itself. This could be code formatting, fixing inconsistent variable names, updating scripts, changes in documentation. ▪ DOCS: (Patch Version) - Changes to documentation `<JIRA_ID>` is the ticket number of the ticket being worked on in this commit. `<SUBJECT>` is a short line of text describing the content of the commit. `[<BODY>]` is an optional commit message body which can include additional information that does not fit in the subject. Examples: ▪ CHORE(CASS-8642): removes println statements ▪ FIX(CASS-8642): changes Api Gateway integration to ALB from GET to ANY method
GIT-03	subject wording	Proposed /Accepted 03.11.2021	The subject of a commit message should read as if there was an introduction "Applying this commit...". This allows for consistent reading of the git log. Examples: ▪ `declares an explicit dependency from WAF association to API stage` ▪ `sets up the integration from RestAPI to our ALB`

CDK Unit Testing

Guideline	Name	Status	Description
CUT-01	Inventorize security relevant resources	Proposed /Accepted 08.12.2021	For security reasons we should keep track of certain objects in our stack, like Security Groups, IAM Roles, IAM Policies etc. For such objects we should add tests so that we are notified in case there is a change or an additional object of that types.

CUT-02	Test environment specific configuration	Proposed /Accepted 08.12.2021	Specific things are configured differently, depending on if we are on DEV, ACCT or PROD envrionment. Such differences must be tested.
CUT-03	Verify prefix is set for names where it matters	Proposed /Accepted 08.12.2021	In order to be able to deploy our stacks multiple times in the same environment, we are using a prefix for all resources that have to be unique in the account (or globally). We must test that that the prefix is applied consistently for all resources that require it.
CUT-04	Verify resource configuration and count	Proposed /Accepted 08.12.2021	We must not only test that specific resources with specific configurations are created in our cloud formation template, but we also need to verify that the number of resources is correct. This can notify us if resources are added by accident.
CUT-05	Keep the unit tests up to date	Proposed /Accepted 08.12.2021	When updating a stack the unit tests of the stack also need to be updated (if not the test coverage is probably too low). **Info:** We did start without testing our CDK code, which means that at the moment of writing this guidelines the tests are still incomplete.
CUT-06	Use descriptive test names	Proposed /Accepted 08.12.2021	The test names should be readable and make clear what is being tested. Use the @DisplayName annotation to have better readable test names.

Java Apps with Lambda

Guideline	Name	Status	Description
JL-01	Keep the lambda handler clean	Proposed /Accepted 08.12.2021	The lambda handler should only be responsible for parsing the invoking event and creating required resources. It should not contain any business logic at all.
JL-02	Cache expensive resources	Proposed /Accepted 08.12.2021	A lambda class is instantiated once but can be called multiple times before the instance is recycled. This means we can reuse certain resources (secrets, db connection etc.) accross invocations. By caching such resources we can improve the execution time of subsequent calls to a lambda function instance.
JL-03	Avoid big libraries	Proposed /Accepted 08.12.2021	Lambda is rather meant to be used for small self contained services than big applications (spring boot). In order to keep the cold start time as low as possible, we need to keep our build as small as possible.

Java Apps with Springboot on Fargate

Guideline	Name	Status	Description
JS-01	Configuration as code	Proposed/Accepted 08.12.2021	