Notes

Serverless Framework and Lambda functions

1. **Additional stack resource limits**
   Only 10 resources is allowed to be added at once via Cloudformation to additional stack. Ex. 10 DynamoDB tables. I recommend splitting them and adding more stacks.

```
additionalStacks:
  databaseTables1:
    Resources:
      #API Entities (models)
      DbTableAppSettings: ${PATH}
      DbTableAccountRole: ${PATH}
      DbTableAccountInvitattion: ${PATH}
      DbTableAccount: ${PATH}
      DbTableCounter: ${PATH}
      DbTableProject: ${PATH}
      DbTableProjectReview: ${PATH}
      DbTableCampaign: ${PATH}
      DbTableAdGroup:${PATH}
      DbTableSite: ${PATH}
  databaseTables2:
    Resources:
      #System models
      DbTableSystemDbMigration: ${PATH}
```

2. **Check if need modules are saved to package.json**
   Serverless framework won't tell you if some module is not installed (because of the fact that lambda is not executed), so you need to take care of this and check all the used and installed modules

3. **Not closed DB connect timeout**
   If DB connection is not closed itself, lambda will wait until the call stack is empty. It will just timeout. This can be configured

DynamoDB

1. **DynamoDB ID**
   DynamoDB doesn't provide a way to set ID
   There's no way to perform entity ID auto create using DynamoDB core. Means that you can setup ID increment to use integer id. It can be done manually.

   I'd recommend using a separate table (Ex. IdTable) to store current/next id for different entities. This will create a new id for a particular entity and increment current value.
   If you want numeric id without conflicts, you can use updateItem Expressions +1. This will provide atomic update and increment counter.

```
const query = {
  TableName: '<TABLE NAME>', // Ex. IdTable
  ReturnValues: 'UPDATED_NEW',
  ExpressionAttributeValues: {
    ':a': 1,
  },
  ExpressionAttributeNames: {
    '#v': 'value',
  },
  UpdateExpression: 'SET #v = #v + :a',
  Key: {
    name: '<KEY NAME>', // Ex. accountId
  },
};
```

   BUT!
   Atomic counters not good for strong requirements (may cause double increment)

   https://docs.aws.amazon.com/en_us/amazondynamodb/latest/developerguide/Worki

ngWithItems.html#WorkingWithItems.AtomicCounters

2. **Streams do not work offline**
   There's no way to run/check DynamoDB streams using offline mode

3. **Scan() is not your friend**
   Scan operation is very slow in DynamoDB.
   600-800к users with read capacity 50 will require ~ 30 minutes to be scanned.
   https://docs.aws.amazon.com/en_us/amazondynamodb/latest/developerguide/bp-query-scan.html

4. **No warning for capacity**
   DynamoDB won't tell you if capacity is overused. Will just not respond.

5. **Best practises**
   https://docs.aws.amazon.com/en_us/amazondynamodb/latest/developerguide/best-practices.html

Algolia Search Engine

1. **Filter by date**
   Doesn't allow to filter by date as an object. Recommend storing dates as a timestamp (Number, not String)

Cloudformation (serverless.yml file)

1. **AWS Cognito Custom attributes can’t be required**
   Required custom attributes are not supported currently. Required custom attributes are not supported currently. There’s no way to add new (not included by default) user attributes and make them required for AWS Cognito validation.

2. **AWS Cognito will be re-created from scratch if you change settings in the file**
   Users fille be removed and AWS Cognito pool will be re-created if you change Cognito configurations. Modifying the schema requires replacement with cloudformation which will delete and recreate your pool.

   The update code isn't advanced enough to detect that you added a user attribute and call the AddCustomAttributes api, it only can modify things accessible to the UpdateUserPool api. If you need to add a new attribute, you should either use the command line or the console to do it if you have previously created the pool.

   UserPoolClient also requires replacement when some attributes are modified.

   http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-cognito-userpool.html#cfn-cognito-userpool-schema

   https://forums.aws.amazon.com/thread.jspa?threadID=259119&tstart=0

   http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-cognito-userpoolclient.html

Deployment

1. **Environments**
   I'd recommend creation AWS Sub-accounts for different environments. This will help to avoid conflicts when using different settings for architecture items and will help to manage an access layer.
   a. Create AWS Master account
   b. Create Administrator user in Master account (define; Administrators group and attach the AdministratorAccess policy to it).
   c. Login to AWS using Administrator user.
   d. Go to My Organization and add new AWS accounts - Member (you have to create new accounts and each account must have a different email address).
   e. Copy the account ID of the Master Account.
   f. Login to the Member account (with the root user).
   g. Create a new Role - Choose Another AWS Account as a trusted entity. Paste the account ID of the Master account and attach AdministratorAccess policy to it. Name the Role admin_member_account.
   h. Copy the account ID of the Member account.
   i. Login with Administrator user to Master account.
   j. Go to Switch Role - add Member account ID in the account field and the role name: admin_member_account in the Role field. Choose color and switch to the member account.

   https://docs.aws.amazon.com/translate/latest/dg/setting-up.html
   https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_accounts_create.html
   https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_accounts_access.html#orgs_manage_accounts_access-cross-account-role

2. **node_modules deployment**
   Not all the modules can be pushed. It doesn't push node_modules, don't even check all the dependencies.
   There's 'force' method to deploy by force

   https://serverless.com/framework/docs/providers/aws/guide/deploying/

Other

1. Cloudwatch: better to stringify JSON's, they will be shown well in CloudWatch UI
2. CircleCI each RUN command creates a new Shell, they don’t share environment
3. CircleCI AWS credentials can't be set in config - hack:
   - run: mkdir ~/.aws
   - run: echo -e "[$**AWS_PROFILE**]\naws_access_key_id=$**AWS_ACCESS_KEY_ID**\naws_secret_access_key=$**AWS_SECRET_ACCESS_KEY**\n" > ~/.aws/credentials
4. CircleCI has a limit of resource usage (builds or something like that) for free version
5. Good middleware library: https://github.com/middyjs/middy