

Najbardziej przydatne komendy AWS CLI dotyczące AWS CloudFormation

Pełna lista znajduje się w [dokumentacji](#) :)

Tworzenie stacków

aws cloudformation create-stack

Tworzymy stack podając lokalizację szablonu oraz parametrów w pliku.

```
aws cloudformation create-stack \  
  --stack name $STACK_NAME \  
  --template-body file://$PATH_TO_MY_TEMPLATE \  
  --parameters file://$PATH_TO_MY_FILE_WITH_PARAMS
```

Jeśli w ramach stacka będziemy tworzyć role lub polityki, potrzebna jest dodatkowa opcja: `--capabilities`. Po niej stosujemy `CAPABILITY_IAM`, `CAPABILITY_NAMED_IAM` lub `CAPABILITY_AUTO_EXPAND`.

```
aws cloudformation create-stack \  
  --stack name $STACK_NAME \  
  --template-body file://$PATH_TO_MY_TEMPLATE \  
  --parameters file://$PATH_TO_MY_FILE_WITH_PARAMS \  
  --capabilities CAPABILITY_NAMED_IAM
```

Dobłą praktyką jest utworzenie dedykowanej roli (np. per projekt), której będziemy używać do tworzenia, aktualizacji i usuwania stacków. Rola ta może mieć pełne uprawnienia, a uprawnionym do jej używania będzie usługa AWS CloudFormation. W takim wariancie osoba (lub aplikacja) wywołująca polecenie `aws cloudformation create-stack` nie musi mieć pełnych uprawnień, a jedynie uprawnienia do tworzenia, aktualizacji i usuwania stacków. W tym rozwiązaniu dodajemy opcję `--role-arn` `$DEDICATED_ROLE_FOR_THIS_STACK`.

```
aws cloudformation create-stack \  
  --stack name $STACK_NAME \  
  --template-body file://$PATH_TO_MY_TEMPLATE \  
  --parameters file://$PATH_TO_MY_FILE_WITH_PARAMS \  
  --capabilities CAPABILITY_NAMED_IAM \  
  --role-arn $DEDICATED_ROLE_FOR_THIS_STACK
```

Do stacka możemy dodać tagi:

```
aws cloudformation create-stack \  
  --stack name $STACK_NAME \  
  --template-body file://$PATH_TO_MY_TEMPLATE \  
  --parameters file://$PATH_TO_MY_FILE_WITH_PARAMS \  
  --capabilities CAPABILITY_NAMED_IAM \  
  --tags Key=Value
```

```
--role-arn $DEDICATED_ROLE_FOR_THIS_STACK \  
--tags Key=$MY_TAG_KEY,Value=$MY_TAG_VALUE  
Key=$ANOTHER_TAG_KEY,Value=$ANOTHER_TAG_VALUE
```

Szablon, z którego tworzymy stack, może znajdować się również w buckecie S3:

```
aws cloudformation create-stack \  
--stack name $STACK_NAME \  
--template-url https://$MY_BUCKET_WITH_TEMPLATES.s3-  
$REGION.amazonaws.com/$MY_TEMPLATE \  
--parameters file://$PATH_TO_MY_FILE_WITH_PARAMS \  
--capabilities CAPABILITY_NAMED_IAM \  
--role-arn $DEDICATED_ROLE_FOR_THIS_STACK \  
--tags Key=$MY_TAG_KEY,Value=$MY_TAG_VALUE  
Key=$ANOTHER_TAG_KEY,Value=$ANOTHER_TAG_VALUE
```

Parametry możemy podawać bezpośrednio w poleceniu:

```
aws cloudformation create-stack \  
--stack name $STACK_NAME \  
--template-body file://$PATH_TO_MY_TEMPLATE \  
--parameters ParameterKey=$MY_PARAM_KEY,ParameterValue=$MY_PARAM_VALUE  
ParameterKey=$ANOTHER_PARAM_KEY,ParameterValue=$ANOTHER_PARAM_VALUE \  
--capabilities CAPABILITY_NAMED_IAM \  
--role-arn $DEDICATED_ROLE_FOR_THIS_STACK \  
--tags Key=$MY_TAG_KEY,Value=$MY_TAG_VALUE  
Key=$ANOTHER_TAG_KEY,Value=$ANOTHER_TAG_VALUE
```

Tworzenie artefaktów

aws cloudformation package

Tego polecenia używamy np. tworząc nested stacks. Przed jego wykonaniem będzie potrzebny dedykowany bucket S3, w którym znajdą się artefakty, czyli gotowe do wdrożenia szablony.

```
aws cloudformation package \  
--template-file $PATH_TO_MY_TEMPLATE \  
--s3-bucket $MY_BUCKET_NAME
```

Nasz spakowany szablon domyślnie znajdzie się w pliku `packaged.yaml`, ale możemy wskazać inny plik:

```
aws cloudformation package \  
--template-file $PATH_TO_MY_TEMPLATE \  
--s3-bucket $MY_BUCKET_NAME \  
--output-template-file $PATH_TO_OUTPUT_TEMPLATE
```

Tworząc stack korzystamy z utworzonego w ten sposób artefaktu.

Aktualizacja stacków

aws cloudformation update-stack

Do aktualizacji stacków używamy polecenia `aws cloudformation update-stack`.

Przykładowe polecenie:

```
aws cloudformation update-stack \  
  --stack-name $STACK_NAME \  
  --template-body file://$PATH_TO_MY_TEMPLATE \  
  --parameters file://$PATH_TO_MY_FILE_WITH_PARAMS \  
  --capabilities CAPABILITY_NAMED_IAM \  
  --role-arn $DEDICATED_ROLE_FOR_THIS_STACK \  
  --tags Key=$MY_TAG_KEY,Value=$MY_TAG_VALUE  
Key=$ANOTHER_TAG_KEY,Value=$ANOTHER_TAG_VALUE
```

aws cloudformation create-change-set

Bezpieczniejszą opcją jest aktualizacja stacków poprzez tworzenie, przegląd i wykonanie tzw. change set.

Najpierw stworzymy change set:

```
aws cloudformation create-change-set \  
  --change-set-name $CHANGESET_NAME \  
  --stack-name $MY_STACK_NAME \  
  --template-body file://$PATH_TO_MY_TEMPLATE \  
  --parameters file://$PATH_TO_MY_FILE_WITH_PARAMS \  
  --capabilities CAPABILITY_NAMED_IAM \  
  --role-arn $DEDICATED_ROLE_FOR_THIS_STACK \  
  --output text \  
  --query [Id] \  
  --tags Key=$MY_TAG_KEY,Value=$MY_TAG_VALUE  
Key=$ANOTHER_TAG_KEY,Value=$ANOTHER_TAG_VALUE
```

Zwróć uwagę na opcję `--query` - pozwala ona z całej odpowiedzi wyświetlić tylko te części, które są interesujące (w tym wypadku będzie to ID change setu, które będzie potrzebne w kolejnych poleceniach)

Następnie przeglądamy change set (możesz to zrobić w CLI albo w konsoli AWS - jak Ci wygodniej) za pomocą polecenia `aws cloudformation describe-change-set`:

```
aws cloudformation describe-change-set \  
  --change-set-name $CHANGESET_NAME \  
  --stack-name $MY_STACK_NAME
```

lub

```
aws cloudformation describe-change-set \  
  --change-set-name $CHANGESET_ARN
```

W odpowiedzi otrzymamy zestaw zmian, jakie nastąpią w stacku.

Akceptacja zmian i wykonanie change setu

Po akceptacji zmian wykonujemy change set:

```
aws cloudformation execute-change-set \  
  --change-set-name $CHANGESET_NAME \  
  --stack-name $MY_STACK_NAME
```

lub

```
aws cloudformation execute-change-set \  
  --change-set-name $CHANGESET_ARN
```

Odrzucenie zmian i usunięcie change setu

Jeśli nie akceptujemy zmian, usuwamy change set:

```
aws cloudformation delete-change-set \  
  --change-set-name $CHANGESET_NAME \  
  --stack-name $MY_STACK_NAME
```

lub

```
aws cloudformation delete-change-set \  
  --change-set-name $CHANGESET_ARN
```

Tworzenie i aktualizacja stacków - w jednym poleceniu

aws cloudformation deploy

Jest to alternatywne polecenie do `aws cloudformation create-stack` i `aws cloudformation update-stack`. Składnia jest nieco inna:

```
aws cloudformation deploy \  
  aws cloudformation deploy \  
  --template-file $PATH_TO_MY_TEMPLATE \  
  --stack-name $STACK_NAME \  
  --no-fail-on-empty-changeset \  
  --parameter-overrides $MY_PARAM_KEY=$MY_PARAM_VALUE  
$ANOTHER_PARAM_KEY=$ANOTHER_PARAM_VALUE \  
  --capabilities CAPABILITY_NAMED_IAM \  
  --role-arn $DEDICATED_ROLE_FOR_THIS_STACK \  
  --tags $MY_TAG_KEY=$MY_TAG_VALUE $ANTOHER_TAG_KEY=$ANOTHER_TAG_VALUE
```

Przedtem wszystkim nie da się podać ścieżki do pliku z parameterami. Jeśli chcemy trzymać wartości parametrów w plikach (co jest wygodne, bo można je przechowywać w repozytorium kodu), problem można rozwiązać używając `jq`:

```
PARAM_FILE="my-file-with-parameters.json"  
PARAMS=$(cat "$PARAM_FILE" | jq -jr  
'map("\(.ParameterKey)=\(.ParameterValue)") | join (" ")')
```

```
aws cloudformation deploy \
  aws cloudformation deploy \
  --template-file $PATH_TO_MY_TEMPLATE \
  --stack-name $STACK_NAME \
  --no-fail-on-empty-changeset \
  --parameter-overrides $PARAMS \
  --capabilities CAPABILITY_NAMED_IAM \
  --role-arn $DEDICATED_ROLE_FOR_THIS_STACK \
  --tags $MY_TAG_KEY=$MY_TAG_VALUE $ANTOHER_TAG_KEY=$ANOTHER_TAG_VALUE
```

Opcja `no-fail-on-empty-changeset` oznacza, że możemy utworzyć “pusty” change set - co ma miejsce przy tworzeniu stacków. Dzięki temu możemy używać tego samego polecenia do tworzenia oraz aktualizacji stacków.

Jeśli aktualizujemy stack, możemy przed faktycznym wdrożeniem przejrzeć change set - dodajemy wtedy opcję `--no-execute-changeset`. Jest to zalecana opcja na środowiskach produkcyjnych.

```
aws cloudformation deploy \
  --template-file $PATH_TO_MY_TEMPLATE \
  --stack-name $STACK_NAME \
  --no-fail-on-empty-changeset \
  --no-execute-changeset \
  --parameter-overrides $PARAMS \
  --capabilities CAPABILITY_NAMED_IAM \
  --role-arn $DEDICATED_ROLE_FOR_THIS_STACK \
  --tags $MY_TAG_KEY=$MY_TAG_VALUE $ANTOHER_TAG_KEY=$ANOTHER_TAG_VALUE
```

Żeby wdrożyć stack, wykonujemy wtedy polecenie `aws cloudformation execute-change-set` podając ID change setu. Jeśli nie akceptujemy zmian, kasujemy change set za pomocą `aws cloudformation delete-change-set`

Kasowanie stacków

`aws cloudformation delete-stack`

```
aws cloudformation delete-stack \
  --stack name $STACK_NAME
```

Jeśli do utworzenia stacka używaliśmy dedykowanej roli, przy kasowaniu stacka należy użyć opcji `--role-arn` i podać ARN tej roli:

```
aws cloudformation delete-stack \
  --stack name $STACK_NAME \
  --role-arn $DEDICATED_ROLE_FOR_THIS_STACK
```

Możemy zachować niektóre zasoby:

```
aws cloudformation delete-stack \
  --stack name $STACK_NAME \
  --role-arn $DEDICATED_ROLE_FOR_THIS_STACK \
```

```
--retain-resources $MY_PRECIOUS_RESOURCE_LOGICAL_ID  
$ANOTHER_PRECIOUS_RESOURCE_LOGICAL_ID
```

Inne

Przeglądanie zdarzeń

Zdarza się, że przy wdrożeniu stacka coś pójdzie nie tak. Wtedy warto przejrzeć zdarzenia (events):

```
aws cloudformation describe-stack-events \  
  --stack name $STACK_NAME
```

Przeglądanie wyników (Outputs)

```
aws cloudformation describe-stacks \  
  --stack-name $MY_STACK_NAME \  
  --output text \  
  --query Stacks[].Outputs[]
```

Walidacja szablonów przed wdrożeniem

```
aws cloudformation validate-template \  
  --template-body file://$PATH_TO_MY_TEMPLATE
```

Jeśli masz zainstalowany linter `cfn-lint`, to dodatkowo szablony można zwalidować za pomocą `cfn-lint -t $PATH_TO_MY_TEMPLATE`. Więcej informacji znajdziesz w dokumentacji: [cfn-lint](#)