Tworzenie zasobów pod aplikację

Zbudowanie wszystkich zasobów będzie się składać z wielu kroków - ważne, żeby wykonać je w odpowiedniej kolejności. Na koniec przetestujemy, czy aplikacja działa prawidłowo.

Żeby uruchomić aplikację “Memy w chmurze” (a ściślej - jej back-end) potrzebujemy następujące zasoby:

1. Buckety S3
- na pliki konfiguracyjne
- na zdjęcia
- na wygenerowane memy
2. Baza danych i miejsce na przechowywanie haseł

Będzie to relacyjna baza danych PostgreSQL w usłudze Amazon RDS. Hasła będziemy przechowywać w usłudze AWS Secrets Manager

3. Uprawnienia dla instancji z aplikacją

…czyli IAM Role z odpowiednimi uprawnieniami

4. Szablon z konfiguracją dla instancji - Launch Template

Launch Template pozwoli uruchamiać wiele instancji EC2 z taką samą konfiguracją. Będzie to bardzo przydatne gdy zaczniemy używać autoskalowania. Tworząc Launch template będzie konieczne podanie AMI ID. Jest to publiczne AMI z zainstalowaną już aplikacją oraz agentem CloudWatch, .

5. Instancja EC2 z aplikacją

Instancję utworzymy z Launch Template.

6. Zmieniony jumphost

Dodatkowo będziemy potrzebować jumphost - z narzędziami i uprawnieniami pozwalającymi na komunikację z bazą danych - oraz NAT Gateway.

Zasoby pod aplikację?

Przygotowanie

NAT Gateway

Na tym etapie NAT Gateway będzie już potrzebna. W przeciwnym razie nie da się uruchomić aplikacji gdyż:

- nie będzie można pobrać plików konfiguracyjnych z bucketu S3 przy starcie instancji

- nie będzie można wysyłać metryk i logów z instancji do usługi AWS CloudWatch
- nie będzie można połączyć się do instancji za pomocą AWS Systems Manager - Session Manager

Uruchom zatem NAT Gateway.

Pliki:

```
memes-generator/network/templates/nat-gateway.yaml
memes-generator/network/commands/deploy-nat-gateway.sh
memes-generator/network/parameters/nat-gateway-a-dev.json
```

Jumphost - nowy lub aktualizacja wcześniej utworzonego.

Poprzednio utworzony jumphost nie zawierał wszystkich uprawnień potrzebnych do połączenia się z bazą danych. Usuń go (wystarczy, że usuniesz stack), a w jego miejsce utwórz nowy.

Pliki:

```
memes-generator/operations/templates/jumphost-db.yaml
memes-generator/operations/parameters/jumphost-db-dev.json
memes-generator/operations/commands/deploy-jumphost.sh
```

Zasoby pod aplikację

Buckety S3

Użyjemy 1 szablonu, ale z różnymi paramtrami.

Pliki:

```
memes-generator/application/templates/bucket.yaml
memes-generator/application/parameters/pictures-bucket-dev.json
memes-generator/application/parameters/memes-bucket-dev.json
memes-generator/application/parameters/configuration-bucket-dev.json
memes-generator/application/commands/deploy-bucket.sh
```

Pliki konfiguracyjne

W buckecie `configuration-bucket` trzeba umieścić plik z konfiguracją agenta usługi AWS CloudWatch: `memes-generator/application/config/cloudwatch-config-memes-generator-dev.json`. Polecenia wysyłające plik znajdują się tu: `memes-generator/application/commands/upload-cw-config.sh`

Ponadto trzeba dodać plik z konfiguracją samej usługi memes-generator dla danego środowiska i regionu: `memes-generator/application/config/.memesconf-dev` Polecenia wysyłające plik znajdują się tu: `memes-generator/application/commands/upload-srv-config.sh`

Zdjęcia

Żeby utworzyć memy, potrzebne jest dodanie kilku zdjęć (aplikacja wybiera losowo z bucketu, dodaje tekst i tworzy mema). Polecenia pobierające zdjęcia i dodające je do bucketu znajdują się tu: `memes-generator/application/commands/upload-pictures.sh`

Baza danych

Następnie tworzymy bazę danych oraz sekrety w AWS Secrets Manager.

Pliki:

```
memes-generator/data/templates/database.yaml
memes-generator/data/parameters/database-dev.json
memes-generator/data/commands/deploy-database.sh
```

Utworzenie dedykowanego użytkownika dla aplikacji

Aplikacja będzie korzystać z innego użytkownika niż główny użytkownik w bazie danych. Trzeba go utworzyć. W tym celu zaloguj się do jumphosta i wykonaj polecenia z tego pliku:

```
memes-generator/data/commands/create-db-app-user.sh
```

Część poleceń to pobranie sekretów i informacji o bazie danych z AWS Secrets Manager. Kolejne to już polecenia wykonywane na bazie danych.

Instancja EC2, Launch Template i uprawnienia

Uprawnienia dla aplikacji na instancji EC2

Do instancji przypiszemy rolę. Tym razem utworzymy ją w oddzielnym stacku.

Pliki:

```
memes-generator/application/templates/application-permissions.yaml
memes-generator/application/parameters/application-permissions-dev.json
memes-generator/application/commands/deploy-application-permissions.sh
```

Launch template

Launch template jest wzorcem, z którego możemy utworzyć wiele instancji o tej samej konfiguracji. Kluczowe elementy launch template to:

- AMI - czyli obraz maszyny wirtualnej
- Uprawnienia dla instancji
- Security Group
- Model cenowy

Launch template może mieć kilka wersji. Tworząc instancję możemy podać konkretny numer wersji, użyć ostatniej wersji albo wersji domyślnej.

Pliki:

```
memes-generator/application/templates/launch-template.yaml
memes-generator/application/parameters/launch-template-dev.json
memes-generator/application/commands/deploy-launch-template.sh
```

Aktualne AMI ID to: `ami-00b18b3deb7f286ca` Jeśli AMI zmieni się (np. na skutek aktualizacji aplikacji albo aktualizacji obrazu), informacja o tym pojawi się w opisie lekcji oraz w grupie kursowej.

Instancja EC2

Teraz możemy już utworzyć instancję. Zwróć uwagę, że jeśli nie podamy wersji Launch Template w pliku z parametrami, zostanie użyta najnowsza wersja Launch Template.

Pliki:

```
memes-generator/application/templates/launch-template.yaml
memes-generator/application/parameters/launch-template-dev.json
memes-generator/application/commands/deploy-launch-template.sh
```

Jeśli wszyskie stacki utworzą się prawidłowo, możemy przejść do testowania.

Wykonywanie tych kroków pół-automatycznie jest dość uciążliwe, dlatego wkrótce zajmiemy się ich automatyzacją.