УДК 004

Развертывание приложения Spring в AWS с помощью Terraform

Еровлева Регина Викторовна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Еровлев Павел Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье будут рассмотрены работы по выполнению развертывания приложения на AWS с использованием Terraform. Работа будет происходить в среде разработки IntelijIdea.
Ключевые слова: Terraform, Spring, Maven

Deploying your Spring boot application to AWS with Terraform

Erovleva Regina Viktorovna
Sholom-Aleichem Priamursky State University
Student

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article will walk you through the work of deploying your application on AWS using Terraform. The work will take place in the IntelijIdea development environment.
Keywords: Terraform, Spring, Maven

Разработка приложения требует нескольких аспектов. Один из этих аспектов - поддерживать инфраструктуру. Традиционно развертывание приложения было ручным процессом, когда системный инженер создавал инфраструктуру, необходимую для запуска кода. За последнее десятилетие цифровой стандарт сильно изменился, как и инфраструктура. Многие люди сейчас запускают приложения в облаке. Эти облачные провайдеры предоставляют сотни услуг, каждая из которых предназначена для решения конкретных задач. Это означает, что настройка инфраструктуры намного сложнее, чем раньше. К счастью, инструментальные средства также претерпели значительные изменения, и теперь существуют некоторые

инструменты. Эти инструменты позволяют описывать инфраструктуру как код. Кроме того, они создадут или разрушат необходимую инфраструктуру.

Terraform - один из таких инструментов. Приятно то, что он поставляется с множеством модулей, которые позволяют настраивать инфраструктуру на многих облачных провайдерах.

Цель данной статьи – развернуть приложение Spring с использованием Terraform.

П.В. Прохоров, Н.В. Разговоров рассмотрели в своей работе современные подходы и технологии в разработке серверных приложений на примере онлайн-магазина с использованием Spring [1]. В своей статье В.С. Жданова описала подход в реализации серверной части клиент-серверного мобильного приложения для просмотра расписания [2]. В своей статье А.И. Егунова, А.А. Аббакумов, М.А. Воропаева, Ю.С. Вечканова рассматривают проблемы повышения эффективности образовательной деятельности вуза и использование научной интеллектуальной собственности в преподавательской деятельности. Так же предложили реализацию информационного хранилища и поисковой системы в виде J2EE-приложения с использованием фреймворка Spring [3]. Так же А.Д. Нарижный, Н.Е. Губенко провели сравнительный анализ технологий, которые имеют схожую функциональность и которые предназначены для одних и тех же задач. Это технологии стеков Spring и JavaEE, которые предназначены для разработки Enterprise-приложений [4]. В.Л. Волушкова рассмотрела в своей работе технологии программирования на примере языка Java [5].

Для начала перейдем на terraform.io, чтобы загрузить Terraform в систему. После установки можем приступить к созданию первого скрипта Terraform. Начнем с создания файла provider.tf. Terraform использует особый язык конфигурации, который называется HCL. Первый шаг - настроить провайдера, которого хотим использовать (рис.1).

```
provider "aws" {
  region = "eu-west-1"
  shared_credentials_file = "$HOME/.aws/credentials"
}
```

Рисунок 1 – Настройка провайдера

Здесь следует отметить несколько моментов. Прежде всего, нужно указать, какой регион хотим использовать на AWS. Регион зависит от нескольких факторов, например, где необходимо хранить данные и где живут клиенты.

Кроме того, необходимо указать Terraform, как войти в AWS. Одна из возможностей - через файл учетных данных. Этот файл учетных данных создается, если входить в систему AWS с помощью интерфейса командной строки AWS.

Чтобы иметь возможность развернуть приложение, сначала нужно где-то сохранить JAR-файл на AWS. Обычно для этого используем конфигурацию S3. Чтобы настроить это, создадим новый файл Terraform с именем main.tf (рис.2).

```
resource "aws_s3_bucket" "s3_bucket_myapp" {
  bucket = "myapp-prod"
  acl = "private"
}
```

Рисунок 2 – Настройка выходного файла

В этом случае создается конфигурация S3 под названием «myapp-dev», которая будет содержать результаты для производственной среды.

Чтобы развернуть приложение, будем использовать Elastic Beanstalk. Elastic Beanstalk — это платформа как услуга (PaaS). Тут идет предоставление приложения, а облачный провайдер предоставляет все остальное.

В скриптах Terraform сначала нужно создать приложение (рис.3).

```
resource "aws_elastic_beanstalk_application" "beanstalk_myapp" {
  name = "myapp"
  description = "The description of my application"
}
```

Рисунок 3 – Настройка развертывания

После этого нужно создать версию, в которой сообщим AWS, где найти приложение (рис.4).

```
resource "aws_elastic_beanstalk_application_version" "beanstalk_myapp_version"
  application = aws_elastic_beanstalk_application.beanstalk_myapp.name
  bucket = aws_s3_bucket.s3_bucket_myapp.id
  key = aws_s3_bucket_object.s3_bucket_object_myapp.id
  name = "myapp-1.0.0"
}
```

Рисунок 4 – Настройка развертывания

Здесь объединяется все, что было сделано. Используя bucket и key свойства, то говорим Terraform/AWS, где найти JAR-файл.

Последняя часть - создать подходящую среду. Чтобы настроить среду, сначала нужно определить имя стека решений. Это особая метка, которая указывает, какую операционную систему и среду выполнения Java хотим использовать (рис.5).

```
resource "aws_elastic_beanstalk_environment" "beanstalk_myapp_env" {
  name = "myapp-prod"
  application = aws_elastic_beanstalk_application.beanstalk_myapp.name
  solution_stack_name = "64bit Amazon Linux 2 v3.1.7 running Corretto 11"
  version_label = aws_elastic_beanstalk_application_version.beanstalk_myapp_version.name


}
```

Рисунок 5 – Настройка среды

Но пока это не будет работать должным образом. Прежде всего, Elastic beanstalk будет проксировать вызовы на порт 5000. Это означает, что нужно запускать приложение на этом порту.

В Spring можно настроить порт, установив SERVERPORT переменную среды (рис.6).

```
resource "aws_elastic_beanstalk_environment" "beanstalk_myapp_env" {
  name = "myapp-prod"
  application = aws_elastic_beanstalk_application.beanstalk_myapp.name
  solution_stack_name = "64bit Amazon Linux 2 v3.1.7 running Corretto 11"
  version_label = aws_elastic_beanstalk_application_version.beanstalk_myapp_version.name

  setting {
    name = "SERVER_PORT"
    namespace = "aws:elasticbeanstalk:application:environment"
    value = "5000"
  }


}
```

Рисунок 6 – Настройка порта

Другой параметр, который нужно настроить — это тип экземпляра. Когда запускаем приложение на Elastic Beanstalk, AWS создает экземпляр EC2. EC2 или Elastic Compute Cloud - еще одна услуга, предлагаемая AWS и обеспечивающая определенную емкость ЦП и памяти.

В данном случае запустим довольно небольшой тип экземпляра под названием «t2.micro» (рис.7).

```
resource "aws_elastic_beanstalk_environment" "beanstalk_myapp_env" {
  name = "myapp-prod"
  application = aws_elastic_beanstalk_application.beanstalk_myapp.name
  solution_stack_name = "64bit Amazon Linux 2 v3.1.7 running Corretto 11"
  version_label = aws_elastic_beanstalk_application_version.beanstalk_myapp_version.name

  setting {
    name = "SERVER_PORT"
    namespace = "aws:elasticbeanstalk:application:environment"
    value = "5000"
  }

  setting {
    namespace = "aws:ec2:instances"
    name = "InstanceTypes"
    value = "t2.micro"
  }

  // ...
}
```

Рисунок 7 – Настройка параметров

И еще один параметр, который нужно настроить, чтобы он работал, - это предоставить профиль IAM. AWS Identity and Access Management или IAM позволяет точно настроить разрешения для доступа к определенным ресурсам (рис.8).

```
resource "aws_elastic_beanstalk_environment" "beanstalk_myapp_env" {
  name = "myapp-prod"
  application = aws_elastic_beanstalk_application.beanstalk_myapp.name
  solution_stack_name = "64bit Amazon Linux 2 v3.1.7 running Corretto 11"
  version_label = aws_elastic_beanstalk_application_version.beanstalk_myapp_version.name

  setting {
    name = "SERVER_PORT"
    namespace = "aws:elasticbeanstalk:application:environment"
    value = "5000"
  }

  setting {
    namespace = "aws:ec2:instances"
    name = "InstanceTypes"
    value = "t2.micro"
  }

  setting {
   namespace = "aws:autoscaling:launchconfiguration"
   name = "IamInstanceProfile"
   value = "aws-elasticbeanstalk-ec2-role"
  }
}
```

Рисунок 8 – Настройка параметра

После настройки среды Elastic Beanstalk можно приступить к ее тестированию. Для этого сначала нужно инициализировать структуру папок Terraform. Это можно сделать с помощью команды: «terraform init».

После этого можно создать ресурсы, выполнив следующую команду: «terraform apply».

Таким образом, было развернуто загрузочное приложение Spring на AWS с помощью Terraform. Также можно легко развернуть загрузочное приложение Spring с помощью AWS CLI или других инструментов. Преимущество использования Terraform возникает, когда нужно управлять дополнительной инфраструктурой, такой как базы данных, другие S3 для приложения и так далее. Поскольку Terraform позволяет обращаться к другим ресурсам, это делает настройку инфраструктуры менее подверженной ошибкам.

Библиографический список

1. Прохоров П.В., Разговоров Н.В. Современные подходы в backend разработке на примере онлайн-магазина // Прикладная математика и фундаментальная информатика. 2020. №2. С. 23-28.
2. Жданова В.С. Серверный модуль системы уведомления об изменениях в расписании занятий // Молодость. Интеллект. Инициатива. 2017. С. 21-22.
3. Егунова А.И., Аббакумов А.А., Воропаева М.А., Вечканова Ю.С. Система

управления хранилищем электронных образовательных ресурсов // Образовательные технологии и общество. 2019. №3. С. 145-154.
4. Нарижный А.Д., Губенко Н.Е. Сравнительный анализ стеков технологий spring и javaee (jakartaee) для разработки enterprise приложений // Информатика, управляющие системы, математическое и компьютерное моделирование 2020. №3. С. 549-462.
5. Волушкова В.Л. Архитектурные решения java для доступа к данным // Теоретические основы программирования. Учебное пособие 2019. 137с.