Professional Skills
Agile Fundamentals
Jira
Git
Databases Introduction
Java Beginner
Maven
Testing (Foundation)
Java Intermediate
HTML
CSS
Javascript
Spring Boot
Selenium
Sonarqube
Advanced Testing (Theory)
Cucumber
MongoDB
Express
NodeJS
React
Express-Testing
Networking
Security
Cloud Fundamentals
AWS Foundations
- AWS Introductions
- AWS Sign up
- AWS Billing Alert
- AWS EC2

Elastic Beanstalk Introduction

Contents

- Overview
- CLI Commands
- Applications
  - Create
  - Show Existing
  - Delete
- Versions
  - Create
    - Sample Version
    - Version Using ZIP or WAR File
  - Show Existing Versions
    - Show All Versions
    - Show Versions for an Application
    - Delete
- Environments
  - Create
  - View Existing
    - For an Application
  - Terminate Environment
- Tutorial

```
  - [Prerequisites](#prerequisites)
```

  - Create the Application
  - Create an Application Version
  - Create an Environment
    - Create a Default VPC
    - Find a Solution Stack
    - Create the Environment
  - Clean Up
- Exercises

Overview

Elastic Beanstalk (EB) on AWS is a Platform as a Service (PaaS) solution for deploying web applications and workloads.
EB can be used with little knowledge of infrastructure, allowing you to focus more on development.

EB has the potential to be used as a quick way to provision a test environment or be used for production solutions with the ability to create different versions to deploy and roll back versions of deployments.

Applications developed with any of the following are supported:

- Java
- .NET
- Nodejs
- Python
- PHP
- Ruby
- Go
- Docker

Common web servers can be also configure in EB:

- Key Pairs
- S3 Introduction
- S3 Storage Options
- AWS S3 bucket creation
- S3 Bucket Policies
- S3 Lifecycle Policies
- S3 File Upload
- S3 AWS-CLI Commands
- S3 Glacier
- Elastic Beanstalk Introduction
- AWS IAM Intro
- AWS IAM User Overview
- AWS IAM Users
- AWS IAM Policies
- AWS Programmatic Access
- AWS IAM Role CLI
- AWS RDS
- AWS Auto-Scaling Group CLI
- Elastic Load Balancer

AWS Intermediate

Linux

DevOps

Jenkins Introduction

Jenkins Pipeline

Markdown

IDE Cheatsheet

- Apache
- Nginx
- Passenger
- IIS

CLI Commands

EB can be easily used through the AWS console; a guided setup is provided. Using the CLI will allow you gain a better understanding of how the different EB components work and the necessary configurations that you need to make. Understanding the CLI commands will also allow you to script solutions for EB and any other resources in AWS for that matter.

More information for the commands that can be used for Elastic Beanstalk can be found here.

Applications

An application is the highest level of configuration for EB.
You may have more than one application in EB if you would like to.

Create

You will need to have a name for your application that isn't the same as any other applications in your EB, you may also provide a description for you application.

```
# aws elasticbeanstalk create-application --application-name [APPLICATION_NAME]
--description "[DESCRIPTION]"
aws elasticbeanstalk create-application --application-name my-first-application
--description "My first EB Application"
```

Show Existing

Existing applications and information about them can be shown by using the `describe-applications` command.

```
# aws elasticbeanstalk describe-applications
aws elasticbeanstalk describe-applications
```

If you have a lot of applications then specific applications can be described using the `--application-names` option:

```
# aws elasticbeanstalk describe-applications --application-names "
[APPLICATION_NAME]" "[APPLICATION_NAME]"
aws elasticbeanstalk describe-applications --application-names "app-1" "app-2"
```

Delete

The application can be deleted by providing the unique name of it to the `delete-application` command:

```
# aws elasticbeanstalk delete-application --application-name [APPLICATION_NAME]
aws elasticbeanstalk delete-application --application-name my-first-application
```

Note before an application can be deleted, there can't be any environments with running code on them.
The easiest way to solve this would be to terminate all the environments beforehand.

Versions

The way that deployments can be managed in EB is through versions of your applications.

Create

There are some required and preferred options to include when creating an application version:

- `--application-name` (required): the name of the application that you are creating a version for
- `--description` (optional): description for the version
- `--source-bundle` (optional): the code to deploy, if ommited, a sample application will be set by AWS. The source bundle can be either from an AWS CodeCommit Git repository or a ZIP or WAR file stored in S3.
- `--auto-create-application` (optional): if you haven't created an application yet then one will be created if you use this option and the application doesn't exist

Sample Version

Here a sample version will be created because a source bundle has not been configured:

```
# aws elasticbeanstalk create-application-version --application-name
[APPLICATION_NAME] --version-label [VERSION_LABEL]
aws elasticbeanstalk create-application --application-name my-first-application
--version-label v1
```

Version Using ZIP or WAR File

The AWS sample applications are good for seeing how EB work however we will of course need to be able to deploy our own code to EB, which can be done using the `--source-bundle` option.
Details on configuring these source bundles can be found in another module.

We will be looking at using a ZIP file for a source bundle in this case which can be retrieved from and S3 bucket.
Note that the S3 bucket that you are using must be in the same region as the EB application.

```
# aws elasticbeanstalk create-application-version --application-name
[APPLICATION_NAME] --version-label [VERSION_LABEL] --source-bundle S3Bucket="
[BUCKET_NAME]",S3Key="[S3_KEY]"
aws elasticbeanstalk create-application --application-name my-first-application
--version-label v1 --source-bundle S3Bucket="my-bucket",S3Key="my-app-v1.zip"
```

Show Existing Versions

Versions of the application that already existing can be viewed by using the `describe-application-versions` command.

Show All Versions

If you provide no arguments then all the versions for every application that you have will be shown.

```
# aws elasticbeanstalk describe-application-versions
aws elasticbeanstalk describe-application-versions
```

Show Versions for an Application

If there is a specific application that you want to see the versions for the `--application-name` option can be provided to filter them out:

```
# aws elasticbeanstalk describe-application-versions --application-name my-
application
aws elasticbeanstalk describe-application-versions --application-name my-
application
```

Delete

A version can be deleted by providing the *application name* and *version label*:

```
# aws elasticbeanstalk delete-application-version --application-name my-
application --version-label [VERSION_LABEL]
aws elasticbeanstalk delete-application-version --application-name my-
application --version-label v1
```

Environments

Environments are completely different instances of the application that are running.
This is of course ideal for testing new features and bug fixes before deploying any new versions to a production instance of the application that is in service.

Create

An environment can be created with the `create-environment` command.
To create a new environment the following information is required:

- Environment Name (`--environment-name`)
  The name for your environment has the following contraints:
  - 4-40 characters in length.
  - Only container letters, numbers and hyphens.
  - Unique within a region in your account - you can't have two environments called `dev` in London for example.
- Application Name (`--application-name`)
  This is the name of the application which has the version that you are wanting to deploy.
- Version Label (`--version-label`)
  This is the specific version that is going to be deployed to the environment.
- Solution Stack (`--solution-stack-name`)
  This is going to be the type of environment that you would like to deploy to - Python, Java etc. Available solution stack names can be found by running `aws elasticbeanstalk list-available-solution-stacks`.
- You must have a default VPC configured in the same region for this application environment, this can be done with `aws ec2 create-default-vpc`
- Some extra options must also be passed in using `--option-settings`, don't worry about the specifics for this for now.

```
# aws elasticbeanstalk create-environment --environment-name [ENVIRONMENT_NAME]
--application-name [APPLICATION_NAME] --version-label [VERSION_LABEL] --
solution-stack-name "[SOLUTION_STACK_NAME]"
aws elasticbeanstalk create-environment --environment-name my-env --application-
name my-app --version-label v1 --solution-stack-name "64bit Amazon Linux 2018.03
v2.9.3 running Python 3.4"
```

View Existing

Existing environments can be viewed using the `describe-environments` command, providing no options will return all environments:

```
# aws elasticbeanstalk describe-environments
aws elasticbeanstalk describe-environments
```

For an Application

You may want to only see the environments for a specific application, this can be done by passing the application name into the `--application-name` argument.

```
# aws elasticbeanstalk describe-environments --application-name
[APPLICATION_NAME]
aws elasticbeanstalk describe-environments --application-name my-app
```

Terminate Environment

Terminating an environment is effectively deleting it which is done by using the `terminate-environment` commmand.
When terminating an environment all you will need to provide is the

environment name:

```
# aws elasticbeanstalk terminate-environment --environment-name
[ENVIRONMENT_NAME]
aws elasticbeanstalk terminate-environment --environment-name [ENVIRONMENT_NAME]
```

Tutorial

Here we are going to deploy an Elastic Beanstalk sample application, which will entail creating:

- An *Application*
- A *Version* of the application to deploy
- An *Environment* to deploy the version of the application in

Prerequisites

- AWS CLI configured

Create the Application

First off we are going to need an application, create one by running the command below:

```
aws elasticbeanstalk create-application --application-name sample-eb-app --
description "Sample Elastic Beanstalk Application"
```

Now at anytime we can see details about the application:

```
aws elasticbeanstalk describe-applications
```

You should see an output like this:

```
{
  "Applications": [
    {
      "ApplicationName": "sample-eb-app",
      "Description": "Sample Elastic Beanstalk Application",
      "DateCreated": "2019-10-19T10:05:53.895Z",
      "ConfigurationTemplates": [],
      "DateUpdated": "2019-10-19T10:05:53.895Z",
      "ResourceLifecycleConfig": {
        "VersionLifecycleConfig": {
          "MaxCountRule": {
            "DeleteSourceFromS3": false,
            "Enabled": false,
            "MaxCount": 200
          },
          "MaxAgeRule": {
            "DeleteSourceFromS3": false,
            "Enabled": false,
            "MaxAgeInDays": 180
          }
        }
      },
      "ApplicationArn": "arn:aws:elasticbeanstalk:eu-west-
2:847151757780:application/sample-eb-app"
    }
  ]
}
```

Create an Application Version

The next thing that is needed is a version of the application which can be deployed, lets create one with a label of `v1` without a source bundle specified so that AWS deploys a sample application for us:

```
aws elasticbeanstalk create-application-version --application-name sample-eb-app
--version-label v1
```

You should now be able to view information about your new application version at any point:

```
aws elasticbeanstalk describe-application-versions
```

You should see something like this:

```
{
  "ApplicationVersions": [
    {
      "ApplicationName": "sample-eb-app",
      "Status": "UNPROCESSED",
      "VersionLabel": "v1",
      "ApplicationVersionArn": "arn:aws:elasticbeanstalk:eu-west-
2:847151757780:applicationversion/sample-eb-app/v1",
      "DateCreated": "2019-10-19T10:11:07.826Z",
      "DateUpdated": "2019-10-19T10:11:07.826Z",
      "SourceBundle": {
        "S3Bucket": "elasticbeanstalk-eu-west-2",
        "S3Key": "GenericSampleApplication"
      }
    }
  ]
}
```

Create an Environment

So this is the last step really, now that we have an application and an application version we can make an environment with the version deployed on it.

Create a Default VPC

Make sure that you have a default VPC available in the same region:

```
aws ec2 create-default-vpc
```

This command might fail, why might that be?
If it does fail that's ok, when it fails its usually because a default VPC exists.

Find a Solution Stack

We will want to know the solution stack name, this can be done by listing the available solution stacks:

```
aws elasticbeanstalk list-available-solution-stacks
```

You will see an output much larger than the example shown below, this has been reduced for the sake of this guide.
Once you have listed the available solution stacks, choose any one of them for your solution stack when we create the environment in the next step.

```
{
  "SolutionStacks": [
    "64bit Amazon Linux 2018.03 v2.9.3 running Python 3.6",
    "64bit Amazon Linux 2018.03 v2.9.3 running Python 3.4",
    "64bit Amazon Linux 2018.03 v2.9.3 running Python",
    "64bit Amazon Linux 2018.03 v2.9.3 running Python 2.7",
    "64bit Amazon Linux 2018.03 v2.11.0 running Ruby 2.6 (Puma)",
    "64bit Debian jessie v2.13.0 running Go 1.3 (Preconfigured - Docker)",
    "64bit Windows Server Core 2012 R2 running IIS 8.5"
  ]
}
```

Create the Environment

We'll call the environment `development` and provide the *application name* and *version label*.
The solution stack name here may not be valid for you so be sure to copy one from when you listed the available solution stacks before.

We will also be passing in options settings for the environment, don't worry about was this does for now, they will be necessary to get this example working however:

```
aws elasticbeanstalk create-environment --environment-name development --application-name sample-eb-app --version-label v1 --solution-stack-name "64bit Amazon Linux 2018.03 v2.9.3 running Python 3.6" --option-settings OptionName="IamInstanceProfile",ResourceName="AWSEBAutoScalingLaunchConfiguration",Namespace="aws:autoscaling:launchconfiguration",Value="aws-elasticbeanstalk-ec2-role"
```

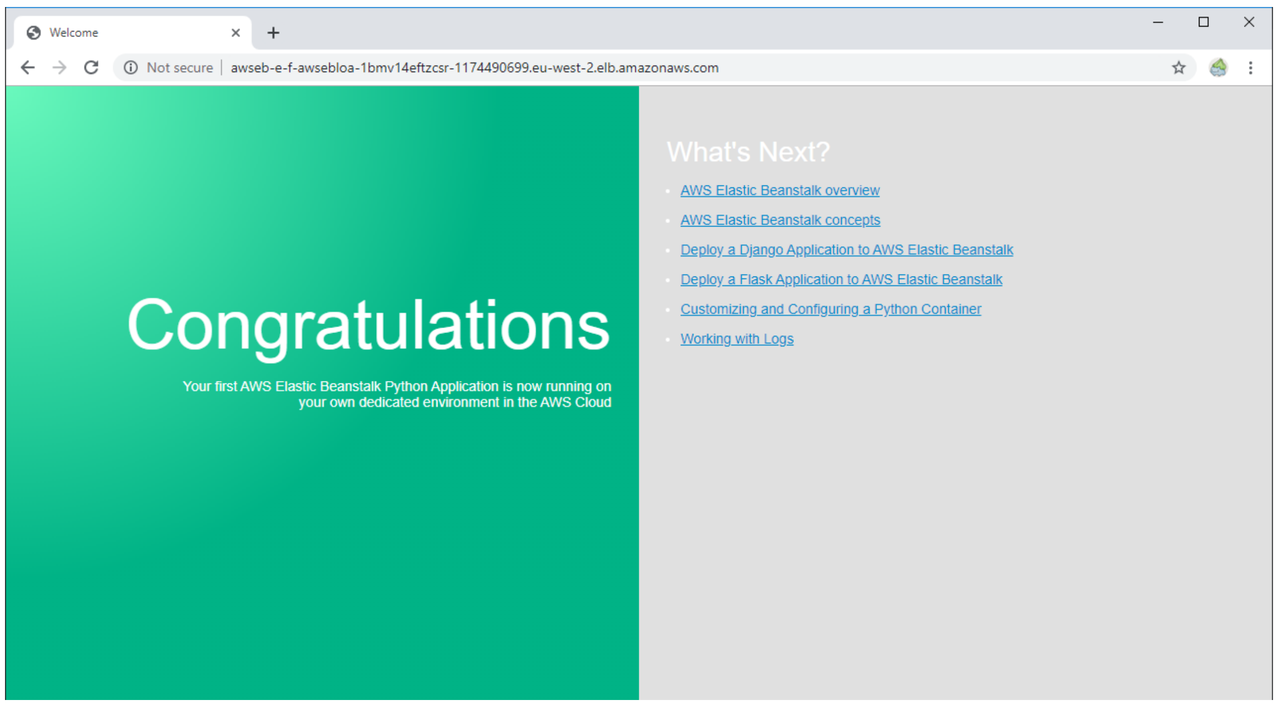
Great, now lets see information about our new environment:

```
aws elasticbeanstalk describe-environments
```

After a few minutes this command should output something like the example below, indicating that the `Status` is `Ready` and there should also be an `EndpointURL` property that you can copy and put in a web browser to access:

```
{
  "Environments": [
    {
      "ApplicationName": "sample-eb-app",
      "EnvironmentName": "development",
      "VersionLabel": "v1",
      "Status": "Ready",
      "EnvironmentArn": "arn:aws:elasticbeanstalk:eu-west-2:847151757780:environment/sample-eb-app/development",
      "EnvironmentLinks": [],
      "PlatformArn": "arn:aws:elasticbeanstalk:eu-west-2::platform/Python 3.6 running on 64bit Amazon Linux/2.9.3",
      "EndpointURL": "awseb-e-f-AWSEBLoa-1BMV14EFTZCSR-1174490699.eu-west-2.elb.amazonaws.com",
      "SolutionStackName": "64bit Amazon Linux 2018.03 v2.9.3 running Python 3.6",
      "EnvironmentId": "e-f9cfpunm2q",
      "CNAME": "development.bdyha2nm2d.eu-west-2.elasticbeanstalk.com",
      "AbortableOperationInProgress": false,
      "Tier": {
        "Version": "1.0",
        "Type": "Standard",
        "Name": "WebServer"
      },
      "Health": "Green",
      "DateUpdated": "2019-10-19T12:14:00.547Z",
      "DateCreated": "2019-10-19T12:11:37.728Z"
    }
  ]
}
```

For this example, once the `EndpointURL` was put in a browser, the sample application can be seen:



awseb-e-f-awsebloa-1bmv14eftzcsr-1174490699.eu-west-2.elb.amazonaws.com

Congratulations

Your first AWS Elastic Beanstalk Python Application is now running on your own dedicated environment in the AWS Cloud

What's Next?

- AWS Elastic Beanstalk overview
- AWS Elastic Beanstalk concepts
- Deploy a Django Application to AWS Elastic Beanstalk
- Deploy a Flask Application to AWS Elastic Beanstalk
- Customizing and Configuring a Python Container
- Working with Logs

Clean Up

We can clean up the resources used in this tutorial by first deleting all the environments and finally the application itself:

```
aws elasticbeanstalk  terminate-environment --environment-name development --
application-name sample-eb-app
aws elasticbeanstalk  delete-application --application-name sample-eb-app
```

Exercises

There are no exerices for this module.