Elastic Beanstalk + Boto3

Tuesday, September 10, 2024 12:57 PM

Table of Contents

Introduction to Elastic Beanstalk

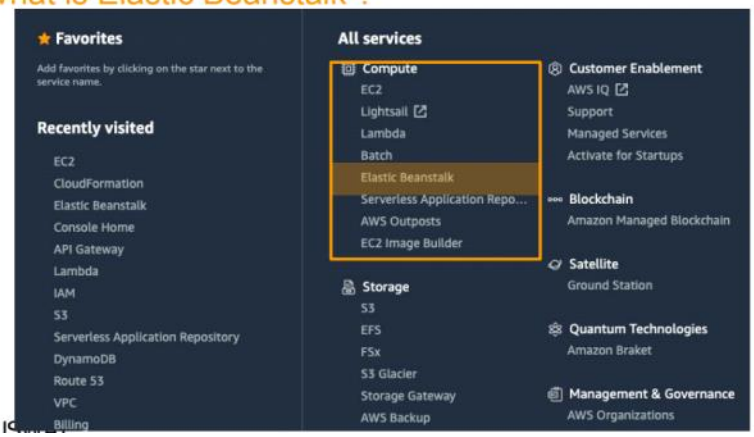
What is Elastic Beanstalk ?

- AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services.
- It is a kind of orchestration service offered by Amazon Web Services used to set up your application architecture.

Cloud engineer kadar na -main servisleri bilen calisana sahip olmayabiliiyor. Kusuk sirketlerde ya da sadece developeri olan sirketlerde developer subnetting olayini bilmek zorunda degil ya da vpcyi, ec2 kaldirirken onun ozelliklerini vs blmek zorunda degil. Ama sirkette bunlari bilen adami istihdam edemiyor.
Developerlarin isi uygulama yartmak. Bizim isimiz bu yaratilan uygulamayi canliya cikarmak.
Developerlar iste bir kodu canliya cikarmak istediklerinde elastic beanstalla bunu yapiyorlar. Eskide opsiyonel kisimlari yoktu daha sade bir yapiydi ama simdi daha ayritni kisimlarini da ekleyebiliyorlar opsiyonel olarak.
Lambda kod yaziyordu, bu ise uygulama.
Yani platform as a service yani PaaS.
COK HIZLI BIR SEKILDE UYGULAMAYI AYAGA KALDIRIYOR.

Introduction to Elastic Beanstalk

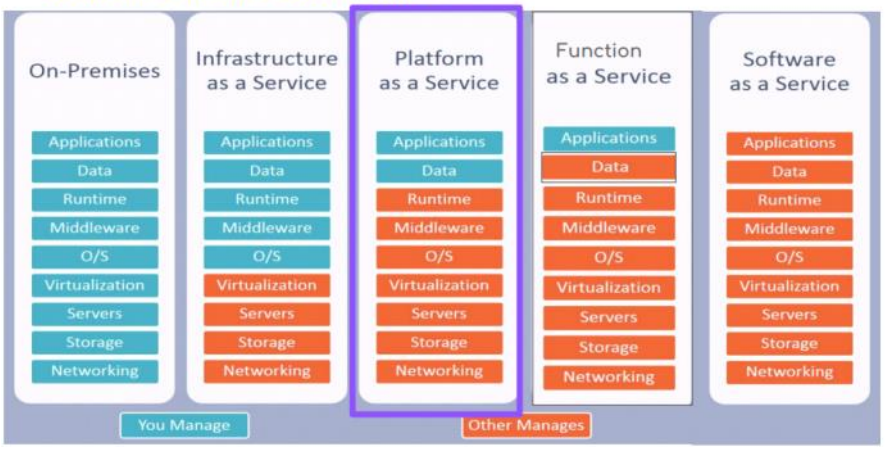
What is Elastic Beanstalk ?



Aplikasyonu kosturacak bir guc olmasi sebeiyle compute altinda yer alir.

Introduction to Elastic Beanstalk

What is Elastic Beanstalk ?



Mesela edevops kisminda maven diye birsey gorecegiz. Orada bu uygulmayi yazdim bitti gibi degilde her ortamda calisabilmesi icin belli belsli kurallari var. iste onlari ziplenmis haliyle veriyorsunuz ve beanstalta onu hayata geciriyor. Buna da PaaS deniyor.

Introduction to Elastic Beanstalk

What is Elastic Beanstalk ?

Go bir environment ya da paython ya da digerleri.
Hatta bilgilsayarlarinizda bir environment yratabiliyorsunuz. O uygulamaya uygun ibraryleri, dependencyleri virtual environmentta indirip sonra silebiliyorsunuz ki boylece bilgisayaranizda kendi lokalinizde yer kaplamamis oluyor. Anlik bir yer kapliyor sonra gidiyor.

Introduction to Elastic Beanstalk

Why AWS Elastic Beanstalk?

- Automates the details of capacity provisioning,
- Load balancing,
- Auto scaling,
- Application deployment,

Bugune kadar gordugumuz bircok servisi beraber kullanip tek tek yapilacaklari belirlemek yerine tek bir yerde toplanmis ve bizde tek tek hizlica ihtiyacimiz olanlari sece sece beanstalkin yaprklarini ister koyup ister cikartabiliyoruz.

Bu aslinda uygulamayi cok hizli bir sekilde deploy etmek amacli ve bunu otomatize ediyor. Otomatize ettigi sey bizim belirlenmis oldugumuz config dosylarla hizli bir sekilde ayaga kalkabiliyor. Versiyonlamayi cok hizli yapabiliyoruz.

Load balancing, autoscaling hep bizim sectigimiz secimler dogrultusunda ayaga kaldiriliyor.

Introduction to Elastic Beanstalk

Why AWS Elastic Beanstalk?

- Automates management tasks:

- Monitoring,
- Version deployment,
- Health check
- Log

Monitoring

Health Check

Autoscaling deyince health check monitoring vs olmzsa olmazimiz.
Version deployment cok kolay bir sekilde gerceklestiriyoruz.

2 Basic Concepts of Elastic Beanstalk

Basic Concepts of Elastic Beanstalk

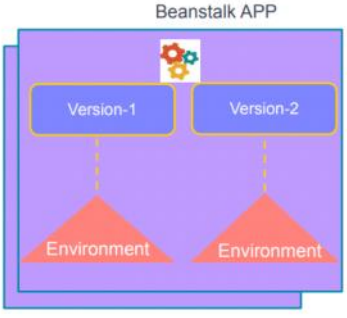
Application

- **Application is a logical collection** of Elastic Beanstalk components. It covers all components.

Application version

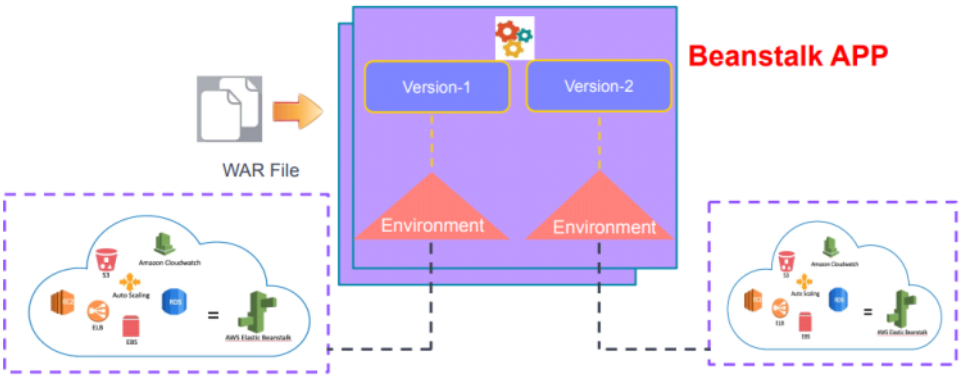
- Specific, labeled iteration of deployable code for a web application.

WAR File



Elastic beanstalkin kendi CLI'i var.
Burada bilmemiz gereken belli basli kavramlar var.
Ilki aplikasyon. Yani uygulama.
Digeri aplikasyonun versiyonu.

Basic Concepts of Elastic Beanstalk



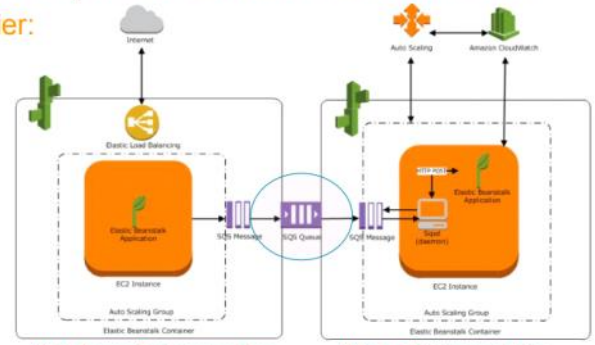
Environment

- An environment is **a collection of AWS resources** running an application version. Each environment runs only one application version at a time.

Aplikasyonlar icinde ortamlar yaratabiliyosunuz.
Bu ortam icersinde en onemli nokta sdece bir uygulamanin bu environmentte calisacagi. Yani iki uygulama ayni environment icinde calisamaz.
Doclerda mesela bir uygulamaya conterilation calistirabilirsin ama ikinciyi yapamazsin.
Test ortami, production ortami, developer ortami vs her ortamda kendisine uygun aplikasyonu calistirirsin.

Basic Concepts of Elastic Beanstalk

Environment Tier:

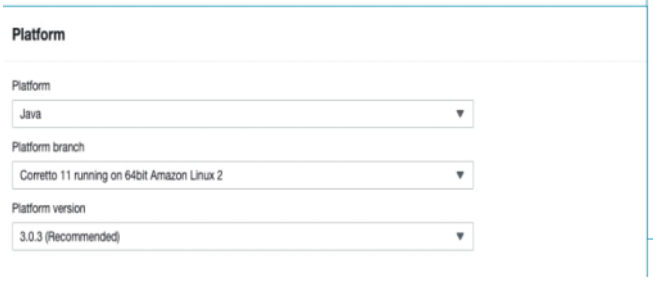


The environment tier designates the type of application that the environment runs, and determines what resources Elastic Beanstalk provisions to support it.

Ortamlarin tierlari var. bor environment dedigimiz kavram var bir de bunun tierlari var.
Bunlar ikiye ayrilir: web server env. Tier ve worker env. Tier
Web server kismi frontend tarafi, worker kismi backend tarafi.
Bu enviromentelerda ne turk ihtiyaclar varsa orada onlar kullaniliyor.
SORULARDA 3 TIER BIR ENVIROMETN AYAGA KALDIRACAGIM DER VE BUNUN UZERINE SORU SORAR!!!!!! BU TREE TIER WORK, WEB, DB.
WEB-FRONTEND(HTTP-HTTPS), WORK-BACKEND(TCP-UDP), DB-DATABASE(MYSQL)
2 TIER DERSE WE-WORKER A DA WEB-DB OLABILIR.WEB-WORK TIER OLARAK GORDUGUMUZDE FRONTEND BACKEND OLARAK DUSUNECEGIZ.

Basic Concepts of Elastic Beanstalk

Platform:



Supported platform versions

- Docker
- Multicontainer Docker
- Preconfigured Docker
- Go
- Java SE
- Tomcat
- .NET Core on Linux
- .NET on Windows Server
- Node.js
- PHP
- Python
- Ruby

Bu eski model. En altta da aplikasyon oluyordu ve calistir diyince bitiyordu. Simdi 5-6 sayfa opsiyonel durumlar var.
Yan taraftaki platformalri destekliyor.

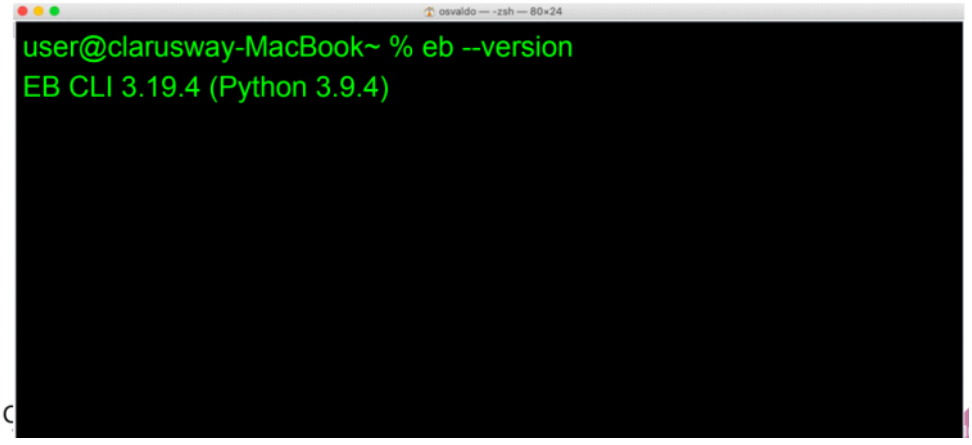
Summary of Terms / Concepts

Concept	What it Means
Application	Logical collection of Elastic Beanstalk components required for a working deployment
Application Version	A labelled version of an application (e.g. 1.0, 1.1, 2.0, etc…)
Environment	A set of AWS resources running a specific application version (e.g. DEV, TEST, PROD)
Environment Tier	The type of application that an environment runs (either Web or Worker)
Platform	Combination of OS, programming language, web server - i.e. the "technology stack"

Bunlar bizim Elastic Beanstalk nezdinde bilmemiz gereken kavramlar.

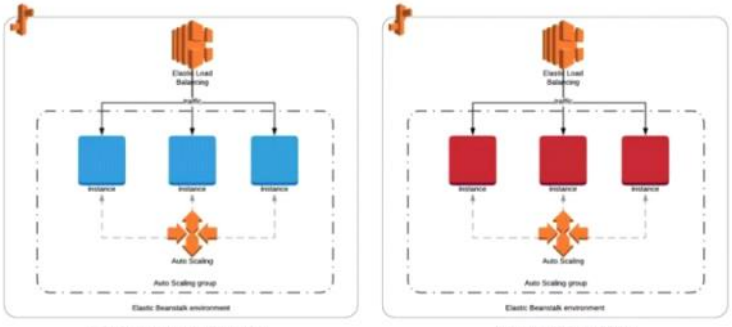
Basic Concepts of Elastic Beanstalk

Elastic Beanstalk Command Line Interface (EB CLI)



Kendi CLI'da sdece beanstalk ozelinde calisbiliyorsunuz. AWS CLI degil de EB CLI olarak gecer.
Is hayatinda CLI ya da oython birini kullanmak zorundasin.

Deployment Models - All at once



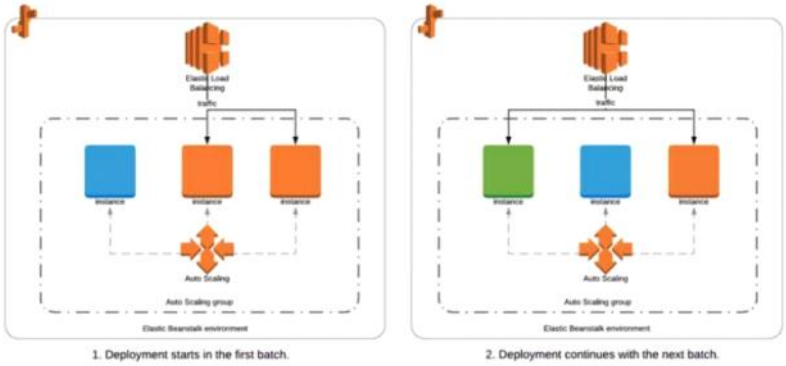
1. Deployment starts in all instances

2. In case of failure, all fails!

Bizim cloud engineer olarak bilmemiz gerekenler deployment modelleri. Deployment yaparken ec2 lari neye gore kaliracak indirecek vs gibi durumlar var. o yuzden mantiksal yapsini bilmemiz gerekiyor.

3 makine kaldirmya kara verdin ve versiyonlama yapacakasin. Ucune de versionlama yaptin calisirsa calsir, calismazsa ucu de birden patlar.
Avantaji cok hizli.
Ekstra bir instancea ihtiyac duymuyor, bu yuzden de maliyet olarak cok uygun.

Deployment Models - Rolling

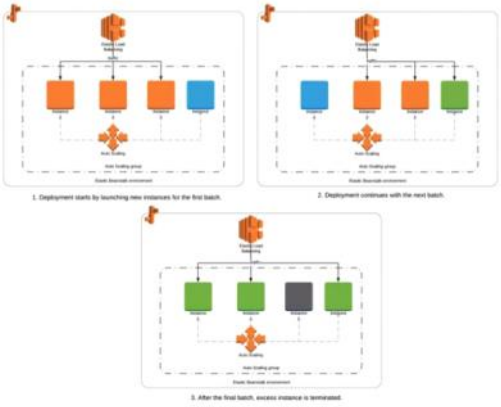


1. Deployment starts in the first batch.

2. Deployment continues with the next batch.

3 makinede trafigi dagitiyor. Rolling secildiginde bir tanesini kesiyor. Kestigine yeni versionu uyguluyor. Eger calisirsa digerini kesiyor ve ona da yukluyor versionu eger o da calisirsa 3. ye geciyor.
Eger calismazsa iki instanceta kalir. Ilkinde patladi calismadi diyelim o zaman iki ve ucuncusune kaldirmaz.
Load balancer olarak yuk dagilimini yapmiyor ama autoscaling olarak 3 instance olarak devam ediyor.
Kullanici kismi patlamis kismi gormez. Bunu biz gormeye devam ediyoruz ve cozmeye calisiyrouz.
Diyelim birinci calisti 2. patladi o zaman duruma gore ilk ve son instancei sirayla gosterir hangisine denk gelirse onu gorur kullanici.

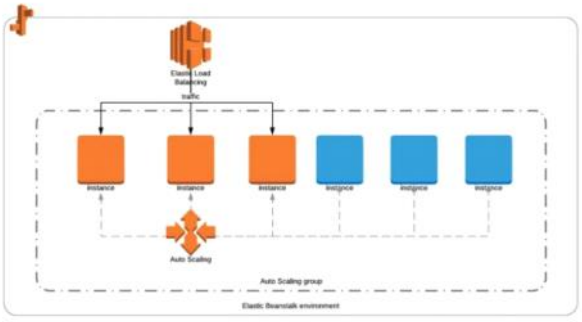
Deployment Models - Rolling with additional batch



Rollong with additional batchte iki instancea dusurmuyor. En az 3 instance kalmasini istiyor.
3 instancetan asagiya dusurmemek icin de bir tane daha ekstra instance uretiyor.
Versiyonlamai ona yukliyor. Eger calisirsa digerine geciyor ve yeni versionu o 3 instancein arasina aliyor. Sonra 2.side calisirsa eskilerden diger 3. bir instancei alip diger yenilenmis 2 versiyonu calistirmaya devam ediyor.
En sona 3 instance ta yenilendikten sonra sona kalani kokten siliyor. Boylece her halukarda elimizde 3 tane instance kalmis oluyor.

ARUSWAY

Deployment Models - Immutable

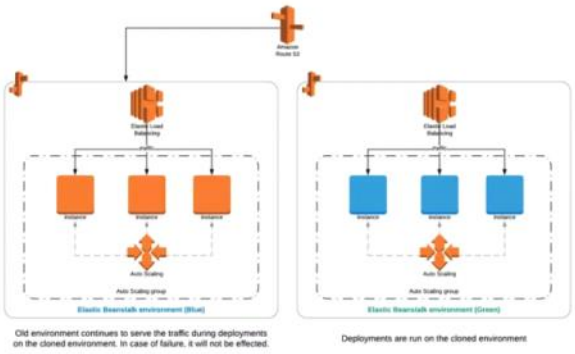


1. Deployment starts by duplicating the instances and deploying the app in new instances

Kac tane instance ihtiyaci varsa o kadari kadar yeni instance olusturuyor tekrar ve onun uzerinde versiyonlama yapip calisirsa diger uc taneyi iptal edip yeni versiyonlanmis instancelari devreye sokuyor. Eger calismazsa her turlu yine de bosa 3 tane instance olusturmus oluyorsun.

Farkettigimiz gibi enduser ya daha az olumsuzluk goruyor ya da hic olumsuzluk gormuyor.
Burada da fiyatlandirma maliyetli oluyor ekstra instance sebebiyle.

Deployment Models - Blue/Green



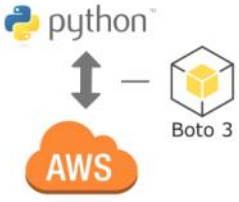
Load balancer tum yapiyi yeni olarak iletiyor.
Yanni load balancer, autoscaling ve instancelari en bastan yaratiyor. Versiyonlamayi yapiyor. Eger hersey okeyse eskiyi kapatip yeniyi devreye sokuyor.
Route 53 burada ne alaka diye dusunursek eger soyle ki domain namei veren route53. Burada koca bir yapi tamamen yok olsa bile kullanici hicirsey gormuyor.

AWS Boto3

CLARUSWAY

Boto3

What is Boto3?



Boto3 is the **AWS SDK** (Software Development Kit) for **Python**. It enables you to create, update, and delete AWS resources with your **Python scripts**. It is basically a **Python library**.

CLARUSWAY

Boto3 daha cok developer icin.
Aws python icin bir softwar development kit yaratirken boto3 kullaniyor.
Developerlarin pythonu kullanarak aws resourcelarina erisme olayi.
Once import boto3 diyoruz. Daha sonra kuralina gore yazip alinan sonuclarina gore kullaniyosun.

Boto3

Installation and Configuration

Installation

pip install boto3 (pip3 install boto3 for Python3)

Access Configuration

aws configure and supply access keys

Boto3

- Boto3 runs on top of botocore, which is foundation for AWS CLI
- A session initiates the connectivity to AWS services. A default session uses the default credential profile(e.g. ~/.aws/credentials, or assume your EC2 using IAM instance profile)

Default session

```
import boto3

# Using the default session
sqs = boto3.client('sqs')
s3 = boto3.resource('s3')
```

Custom session

```
import boto3
import boto3.session

# Create your own session
my_session = boto3.session.Session()
```

Boto3
Client vs Resource

Client

- Low-level AWS service access
- Exposes botocore client to the developer
- Supports all AWS service operations

```
s3 = boto3.client('s3')
```

Resource

- Higher-level, object oriented API
- Exposes sub processes of AWS resources
- Does not provide 100% coverage of AWS API

Iki tane yazim yontemi var. client ve resource.
Clientle yazdiginiz zaman awsin beynine erismis oluyorsun.
Resourcelara ise API uzerinden ulasiyor.
Yazim uzerinden bakinca clientin yazilimi biraz daha zor ama tum servislere erisim var.
Resource ise okumasi yazmsi cok daha kolay ama sinirli erisim. Belirli erisim var. eger bizim kullanacagimiz servisler resourse kapsaminda ise esource ile yazmak daha kolay.

Boto3
Using Boto3

```
import boto3

# Use Amazon S3
s3 = boto3.resource('s3')

# Print out all bucket names
for bucket in s3.buckets.all():
    print(bucket.name)
```

Boto3nin amazons3 servisini kullanacak burada. Ama reource olarak.
Kodlarin hepsi sunu soyluyor bize.var olan tum bucketlari bize listeli.

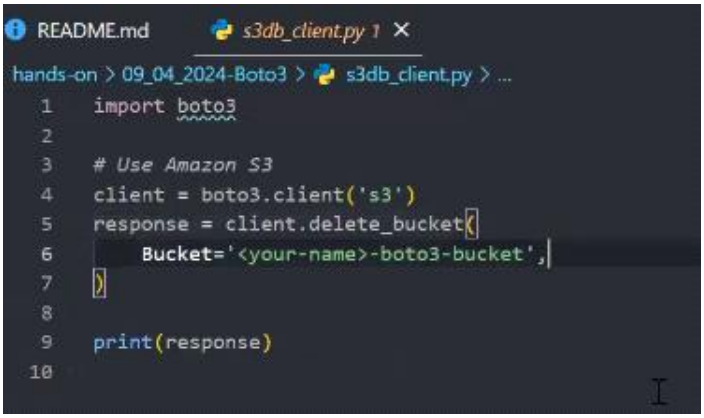
Boto3
Using Boto3

```
import boto3

# Use Amazon S3
s3 = boto3.resource('s3')

# Upload a new file
data = open('test.jpg', 'rb')
s3.Bucket('my-bucket').put_object(Key='test.jpg', Body=data)
```

Burada da yine s3 kullan.
Bir tane data ver verdigi icerikte.
Son olarakta s3 bucketa belirttigi objeyi put et.



Clientle yaptiginda once bir response belirlemen gerekiyor. Ayni json formati gibi.