



# CloudFormation (CFN)


Infrastructure as Code (IaC) bir servistir. Aynı şekilde terraform da IaC'dir.

Oluşturduğumuz belirli bir template'e göre ilgili servisleri ayağa kaldırır. EC2, ASG, ELB vs. Biz toplamda 1-2 servis oluştururken sıkıntı yaşamayız ancak bazen bir mimaride yüzlerce servis oluyor. Bunu CFN ile kolaylıkla yapabiliriz. Ücreti yok ancak kullanılan içinde servisler ücretli.

Oluşacak bu mimariye **stack** denir. CFN'in iki yapısı vardır. Biri stack diğeri de bu stackleri nasıl oluşturacağını gösteren template'dir.

Yazılan bu template'ler YAML ya da JSON formatında yazılır. Bu iki formatta veri tutmaya yarar. Her ikisi de kolayca birbirine çevirilir. ([json2yaml.com](https://json2yaml.com))


## İlgili VS Code extension'ları

 marketplace.visualstudio.com

#### YAML - Visual Studio Marketplace

Extension for Visual Studio Code - YAML Language Support by Red Hat, with built-in Kubernetes syntax support




 marketplace.visualstudio.com

#### YAML ❤️ JSON - Visual Studio Marketplace

Extension for Visual Studio Code - Easily convert yaml to json and json to yaml




 marketplace.visualstudio.com

#### CloudFormation Linter - Visual Studio Marketplace

Extension for Visual Studio Code - AWS CloudFormation template Linter



 marketplace.visualstudio.com

#### CloudFormation Snippets - Visual Studio Marketplace

Extension for Visual Studio Code - This extension adds snippets for all the AWS CloudFormation resources into Visual Studio Code.



## YAML

Daha çok konfigurasyon için kullanılır. Kubernetes, Ansible vs. için bu dosyaları yazıyoruz.

Eskiden XML kullanılıyordu. Sonra YAML ve JSON piyasayı domine etti. Okunuşu kolay, integer ve string değerlerini tutabiliyoruz.

Veriler key-value şeklinde tutulur. Key değerleri string olmak zorunda. Indentation'lar önemli. Hata olmaması adına girintilerde tab yerine iki boşluk kullanılması tavsiye edilir.

Aşağıdaki YAML dosyasında comment yazabiliyoruz. Her bir YAML dosyası 3 tire (---) ile ayrılır. Bunları istersek farklı .yaml dosyalarında da yazabiliriz. Aşağıdaki dosya için 2 farklı stack oluşturur.

```

26 | set of resources
27
28 Outputs:
29 | set of outputs
30
31 ---
32
33 ---
34 AWSTemplateFormatVersion: "version date"
35
36 Description:
37 | String
38
39 Metadata:

```

Listeler tire (-) ile belirtilir.

```

# This is the comment line
---
key: value
instructor: charlie |

---
- charlie
- guile
- osvaldo
- adam

---

[charlie, guile, osvaldo, adam]

```

Pipe (|)'ın anlamı ardından gelen komutları satır satır icra ettirmeye yarar. Örneğin bu pipe'lar user data yazarken kullanılır. Çünkü EC2 ayağa kalkarken user datadaki komutlar tek tek icra ediliyor. Pipe sayesinde bu user dataları CloudFormation içine yazabiliyoruz. yapabiliyoruz.

```

- |
  curl --request PUT "https://gitlab.nioyatech.com/api/v4/groups/201/variables/KUBE_GATEWAY_IMAGE_ECR" \
  --header "PRIVATE-TOKEN: $API_TOK" \
  --form "value=$IMAGE_TAG_ECR"

```

Büyükdür işareti (>), kendinden sonra gelen satırlar ayrı da olsa bunları tek bir komut olarak görüyor.

```
pragraph: > # This text will be a sentence
  this
  line
  will
  be
  in
  single
  line.
```

YAML'da dosyayı yazarken başka bir yere referansta bulunabiliyoruz. Örneğin henüz oluşmamış EC2'nun ID'sini bilmiyoruz. Oraya referansta bulunuyoruz. EC2 ayağ kalktıktan sonra o boşluğu kendi dolduruyor. !Ref ve !GetAtt ile bu referanslamalar oluyor.

## JSON

Data-serialization yapmaya yarar. Nedir bu. Hazırladığımız programları başka bir yerde de kullanmak istiyorsak bu paylaşım olayına denir. Bunu YAML ile de yapabiliriz ancak API'ler arası veri transferinde daha çok JSON kullanılır. Server'lar, programlar birbirleri ile bu iki formatta anlaşır. Çünkü bir çok dil var hepsi en sonunda bu iki formata çevrilip iletilir.

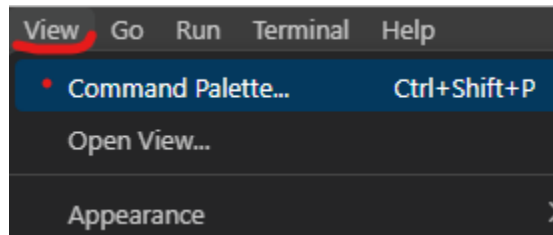
## JSON

```
1 {  
2   "json": [  
3     "rigid",  
4     "better for data interchange"  
5   ],  
6   "yaml": [  
7     "slim and flexible",  
8     "better for configuration"  
9   ],  
10  "object": {  
11    "key": "value",  
12    "array": [  
13      {  
14        "null_value": null  
15      },  
16      {  
17        "boolean": true  
18      },  
19      {  
20        "integer": 1  
21      }  
22    ]  
23  }  
24 }
```

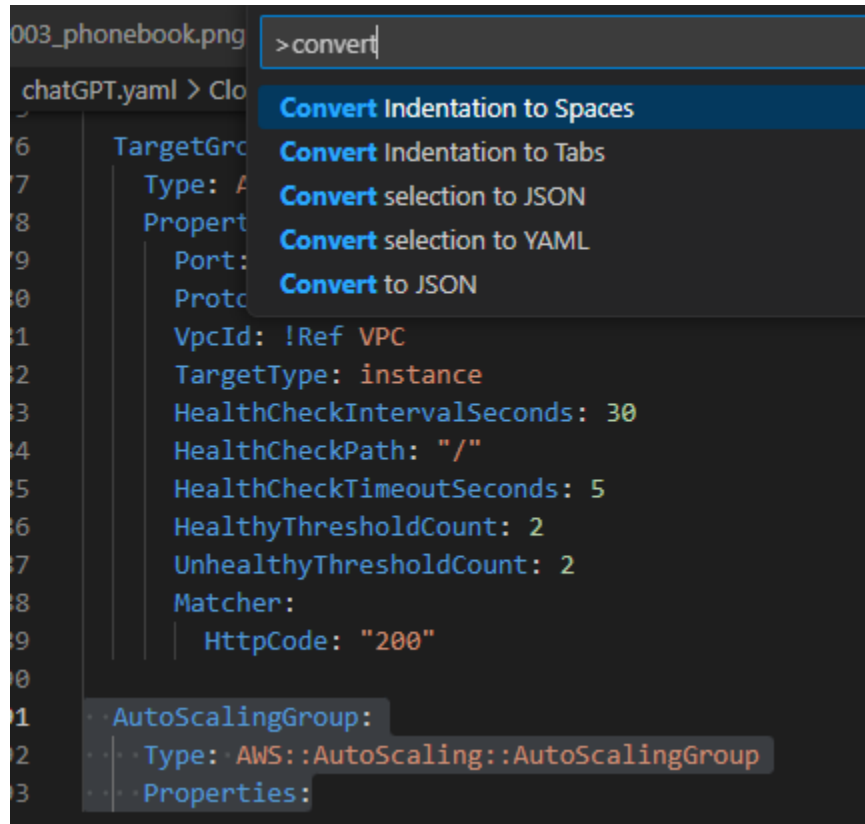
Kiyaslama yaparsak;

Type	YAML	JSON
Comments	Denoted with a hash/number sign	Not allowed
Hierarchy	Mappings, and sequences can be nested. Hierarchy is determined by the indentation level	Objects and arrays can be nested, and are denoted by braces and brackets, respectively.
Arrays	[first, second, 3]	["first", "second", 3]
Strings	Does not require quoting but supports both single and double quotes	Must be double-quoted. Allows character (tabs, newlines, etc.) escaping with a backslash as the escape character.
Numbers	Built-in support for integers, floating-point, octal and hexadecimal numbers	Floating point numbers in scientific notation. Infinity is not permitted.
Date/Timestamp	Supported	Not supported

Yukarıda yüklediğimiz extentionlar sayesinde JSON ve YAML formatlarını birbirine çevirebiliyoruz. Çevirilecek yeri seçip ctrl+shift+p ile command Palette açılır



Açılan panele “convert” yazarsak çeviri seçenekleri karşımıza çıkacaktır.



## TEMPLATES

Oluşacak templateler JSON veya YAML dosyası olmalı. Uzantısı .json, .yaml, .yml, .template, .txt olabilir. Yapısı aşağıdaki gibidir. Resource kısmı doldurulması zorunlu alandır.

- Template Version
- Description
- Metadata
- Parameters
- Mappings
- Conditions
- Transform
- Resources
- Outputs

Bu template'i kendi localimizden de yükleyebiliriz. S3'ten de çekebiliriz.

#### Specify template

A template is a JSON or YAML file that describes your stack's resources and properties.

##### Template source

Selecting a template generates an Amazon S3 URL where it will be stored.

☒ Amazon S3 URL

☐ Upload a template file

##### Amazon S3 URL

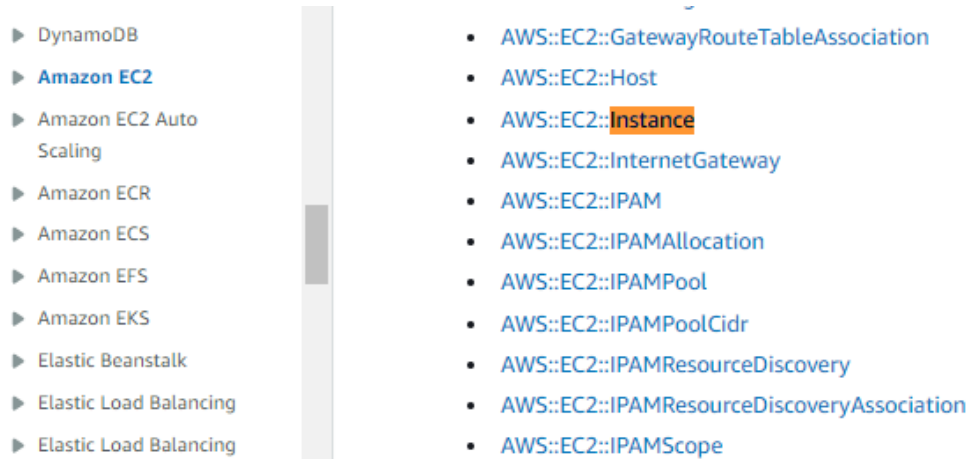
Amazon S3 template URL

Bunun formatını docs.aws'den bulabiliriz. [cloudformation>User Guide>Template reference>Resource and Property Reference](#) kısmından kullanabileceğimiz tüm



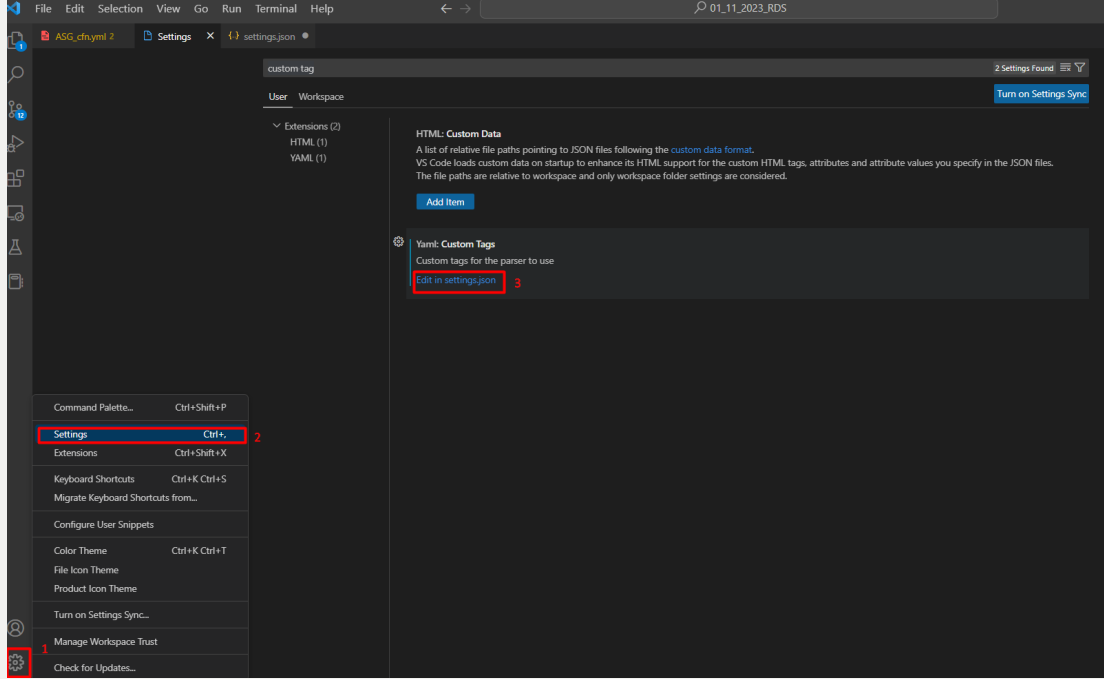
servislere ait template örneklerine ulaşabiliriz.

Örneğin EC2 menüsüne girdiğimizde EC2 ile alakalı tüm alt birimler açılıyor. Instance'ı nasıl ekleyeceğimize bakmak için sağdan instance menüsüne geldik.





Template'i oluřturduktan sonra iinde hata olmasa bile kırmızı izgi ile sanki hata varmıř gibi gsteriliyor. Bunu ařağıdaki gibi dzeltebiliriz.



```
"terminal.integrated.enableMultiLinePasteWarning": false,
"window.zoomLevel": -1,
"workbench.iconTheme": "material-icon-theme",
"terminal.integrated.defaultProfile.windows": "Command Prompt",
"redhat.telemetry.enabled": true,
"editor.accessibilitySupport": "on",
"yaml.customTags": [
  "!Base64 scalar",
  "!Cidr scalar",
  "!And sequence",
  "!Equals sequence",
  "!If sequence",
  "!Not sequence",
  "!Or sequence",
  "!Condition scalar",
  "!FindInMap sequence",
  "!GetAtt scalar",
  "!GetAtt sequence",
  "!GetAZs scalar",
  "!ImportValue scalar",
  "!Join sequence",
  "!Select sequence",
  "!Split sequence",
  "!Sub scalar",
  "!Transform mapping",
  "!Ref scalar",
]
}
```

buradaki yaml.customTags'in  
altına aşağıdaki tagleri  
ekliyoruz kaldırıyoruz

## STACKS

Bu stack ile tek bir EC2'dan oluşan sistem de kurulabilir. Complex bir VPC'de kurabiliriz. Eğer template'de bir hata varsa oluştururken Roll-back hatası alırız.

