

# JSON

- This a file format represneting data
- The data is represented in the form of name value pairs
- syntax

```
'<name>': <value>
```

- type of data
  - simple
    - text
    - number
    - boolean
  - complex
    - object/map/dictionary
    - list
- text: text is represented in quotes (single or double)

```
"model": "Dell Lattitude 7027"
```

- number: number is represented without quotes

```
"price": 190009.99
```

- boolean: this has only two values true or false

```
"fingerprint": true
```

- object: object denoted with a `{}`

```
"address" : {  
  "flatno": "601-A",  
  "building": "Nilgiri"  
}
```

- list:

```
"colors": ["black", "white"]
```

- Birth Certificate in json

```
{
  "DateOfBirth": "12/05/2024",
  "Sex": "Female",
  "Name": "",
  "FathersName": "xxxyx",
  "MothersName": "xxxxx",
  "PlaceOfBirth": {
    "locationType": "Hosptial",
    "Address": {
      "Doorno": "1234",
      "street": "1232",
      "state": "Telangana",
      "pincode": "500016"
    }
  },
  "NumberOfCopies": 2,
  "CourierDelivery": true,
  "MobileNumber": "9999999999"
}
```

- Same in YAML

```
---
DateOfBirth: "12/05/2024"
Sex: "Female"
Name: ""
FathersName: "xxxyx"
MothersName: "xxxxx"
PlaceOfBirth:
  locationType: "Hosptial"
  Address:
    Doorno: "1234"
    street: "1232"
    state: "Telangana"
    pincode: "500016"
NumberOfCopies: "2"
CourierDelivery: "true"
MobileNumber: "9999999999"
```

- Structure we will mention types (grammar|syntax...)

```
{
  "DateOfBirth": text,
  "Sex": text,
  "Name": text,
```

```

"FathersName": text,
"MothersName": text,
"PlaceOfBirth": {
  "locationType": text,
  "Address": {
    "Doorno": text,
    "street": text,
    "state": text,
    "pincode": text
  }
},
"NumberOfCopies": number,
"CourierDelivery": boolean,
"MobileNumber": text
}

```

- Json files are generally created with extensions `.json` and yaml files with extension `.yaml` or `.yml`

## Infrastructure as Code (IaC)

- This approach is about declaring infra as code i.e. we mention what we want
- the stuff which you have declared is called as **desired state** and what is actually present on the cloud is **actual state**
- The difference between desired state and actual state is called as **drift**
- IaC Tools will take desired state in the form of some file i.e. cloud formation takes the json or yaml file as input and ensures your desired state is maintained
- To express the desired state using cloudformation we would be using visual studio code

The screenshot displays the Visual Studio Code interface with the 'EXTENSIONS: MARKETPLACE' view open. The search results for 'cloudformation' are shown, with the top result 'CloudFormation' by 'aws-scripting-guy' circled in green. The extension details panel on the right shows the extension is installed globally. Below the main extension details, there are sections for 'DETAILS', 'FEATURES', 'CHANGELOG', and 'DEPENDENCIES'. The 'News' section mentions 'Introducing two release channels: 1. Stable for scheduled update cycle, 2. Nightly for fast updates based on community commits'. A link to the GitHub releases page is provided: <https://github.com/aws-scripting-guy/cform-VSCode/releases>.

## Cloudformation

- Cloudformation is an Iac Tool offered by AWS.
- Concepts:
  - Template: This is the desired state expressed by us in json or yaml format
  - Stack: when we want execute the template we create stack
- Cloudformation stack will execute the template to create the desired state. In case of any errors it will be reverted back to previous state.
- Cloudformation templates have a structure specified [Refer Here](#)
  - JSON [Refer Here](#)
  - YAML [Refer Here](#)
- In the Cloudformation Templates, Resources is the mandatory section as it describes the desired state.
- The minimal structure

```
{  
  "Resources": {  
  }  
}
```

- To define a Resource [Refer Here](#)

```
{  
  "Resources": {  
    "<logical-id>": {  
      "Type" : "Resource type",  
      "Properties" : {  
      }  
    }  
  }  
}
```