Ch 1 LAB #1 Template Anatomy (in JSON and YAML)

```
{
    “AWSTemplateFormatVersion”
:   “version date”,
    “Description” : “JSON
string”,
    “Metadata” : {
        “template metadata
    },
    “Parameters” : {
        set of parameters
    },
    “Mappings” : {
        set of mappings
    },
    “Conditions” : {
        set of conditions
    },
    “Transform” : {
        set of transforms
    },
    “Resources” : {
        set of resources
    },
    “Outputs” : {
        set of outputs
    }
}
```

AWSTemplateFormatVersion:
“version date”
Description:
    string
Metadata:
    template metadata
Parameters:
    set of parameters
Mappings:
    set of mappings
Conditions:
    set of conditions
Transform:
    set of transforms
Resources:
    set of resources
Outputs:
    set of outputs

AWSTemplateFormatVersion: “version date”
The AWS CloudFormation template version that the template conforms to.
#This is a single line comments
#this
#is a multiple
#line comment
By the way, comments are only allowed in a YAML template not a JSON template.

Description:
Use the description to explain the purpose of the template. In other words, what is the template designed to create? It is written as a string and if you do include a description than it must come after the AWSTemplateFormatVersion
    String

Metadata:
Objects that provide additional information about the template
    template metadata

Parameters:
Values to pass to your template at runtime (when you create or update a stack)

Common things we would need to specify in parameters would be a key pair, or perhaps a VPC for which to launch an EC2 instance in. If we need to provide any details upfront to the template, it would be in the parameters section.
set of parameters

Mappings:
A mapping of keys and associated values that you can use to specify conditional parameter values, similar to a lookup table. In the template that we will build we will have mappings for a region to an AMI ID, because AMI IDs are different depending on the region you are launching in. With the mapping, when we are building our stack we can determine the region that the stack is being created in and then go to our mappings table/aka lookup table and determine the appropriate AMI ID.
You can match a key to a corresponding value by using the Fn::FindInMap intrinsic function.
set of mappings

Conditions:
Conditions that control whether certain resources are created during stack creation. For example, let's say you have a test environment and a production environment (and maybe you want smaller servers for your test environment). So, based on whether you are building a stack for testing or for prod, you can create a parameter where the user selects either test or prod, and this variable is passed into our template and based on that selection will provision test hardware or prod hardware.
set of conditions

Transform:
For serverless applications (also referred to as Lambda-based applications)
Here we specify the version of the AWS Serverless Application Model (AWS SAM) We can have (SAM) code within our template and there is a translation going on.
set of transforms

Resources:
Specifies the stack resources and their properties (Such as EC2 instances, VPCs, security groups, subnets, NACLs, RDS databases, etc) The resources section is where are infrastructure goes. It is the only required element in the cloudformation template.
set of resources

Outputs:
Describes the values that are returned whenever you view your stack's properties. When we build a stack we will have an outputs tab and anything that we have defined in our template in the outputs section will be displayed in our outputs tab in our stack. A good example of this would be that we are building a webserver and we want to provide a url for that webserver. That URL can be put in our outputs section.
Set of outputs

Here is a full template file which launches an EC2 instance. Copy that YAML code below and save the file in a code editor of your choice (such as Brackets or VSC). Save the file as TemplateAnatomy.yaml

```
AWSTemplateFormatVersion: 2010-09-09
Description: >-
  This template creates an EC2 instance based on the region and selection of an
  AMI ID. It also will create a Security Group.
Parameters:
  MySubnet:
    Description: My subnet from my VPC
    Type: String
    Default: subnet-YYYYYYYY
  MySG:
    Description: My Security Group from my VPC
    Type: String
    Default: SG-YYYYYYYY
  KeyName:
    Description: Name of an existing EC2 KeyPair to enable SSH access to the instance
    Type: 'AWS::EC2::KeyPair::KeyName'
    ConstraintDescription: must be the name of an existing EC2 KeyPair.
  InstanceType:
    Description: WebServer EC2 instance type
    Type: String
    Default: t2.small
    AllowedValues:
      - t1.micro
      - t2.nano
      - t2.micro
      - t2.small
      - t2.medium
      - t2.large
      - m1.small
      - m1.medium
      - m1.large
      - m1.xlarge
      - m2.xlarge
      - m2.2xlarge
      - m2.4xlarge
      - m3.medium
      - m3.large
      - m3.xlarge
      - m3.2xlarge
```

```
    - m4.large
    - m4.xlarge
    - m4.2xlarge
    - m4.4xlarge
    - m4.10xlarge
    - c1.medium
    - c1.xlarge
    - c3.large
    - c3.xlarge
    - c3.2xlarge
    - c3.4xlarge
    - c3.8xlarge
    - c4.large
    - c4.xlarge
    - c4.2xlarge
    - c4.4xlarge
    - c4.8xlarge
    - g2.2xlarge
    - g2.8xlarge
    - r3.large
    - r3.xlarge
    - r3.2xlarge
    - r3.4xlarge
    - r3.8xlarge
    - i2.xlarge
    - i2.2xlarge
    - i2.4xlarge
    - i2.8xlarge
    - d2.xlarge
    - d2.2xlarge
    - d2.4xlarge
    - d2.8xlarge
    - hi1.4xlarge
    - hs1.8xlarge
    - cr1.8xlarge
    - cc2.8xlarge
    - cg1.4xlarge
  ConstraintDescription: must be a valid EC2 instance type.
SSHLocation:
  Description: The IP address range that can be used to SSH to the EC2 instances
  Type: String
  MinLength: '9'
```

```
    MaxLength: '18'
    Default: 0.0.0.0/0
    AllowedPattern: '(\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})/(\d{1,2})'
    ConstraintDescription: must be a valid IP CIDR range of the form x.x.x.x/x.
Mappings:
  AWSInstanceType2Arch:
    t1.micro:
      Arch: HVM64
    t2.nano:
      Arch: HVM64
    t2.micro:
      Arch: HVM64
    t2.small:
      Arch: HVM64
    t2.medium:
      Arch: HVM64
    t2.large:
      Arch: HVM64
    m1.small:
      Arch: HVM64
    m1.medium:
      Arch: HVM64
    m1.large:
      Arch: HVM64
    m1.xlarge:
      Arch: HVM64
    m2.xlarge:
      Arch: HVM64
    m2.2xlarge:
      Arch: HVM64
    m2.4xlarge:
      Arch: HVM64
    m3.medium:
      Arch: HVM64
    m3.large:
      Arch: HVM64
    m3.xlarge:
      Arch: HVM64
    m3.2xlarge:
      Arch: HVM64
    m4.large:
      Arch: HVM64
```

m4.xlarge:
  Arch: HVM64
m4.2xlarge:
  Arch: HVM64
m4.4xlarge:
  Arch: HVM64
m4.10xlarge:
  Arch: HVM64
c1.medium:
  Arch: HVM64
c1.xlarge:
  Arch: HVM64
c3.large:
  Arch: HVM64
c3.xlarge:
  Arch: HVM64
c3.2xlarge:
  Arch: HVM64
c3.4xlarge:
  Arch: HVM64
c3.8xlarge:
  Arch: HVM64
c4.large:
  Arch: HVM64
c4.xlarge:
  Arch: HVM64
c4.2xlarge:
  Arch: HVM64
c4.4xlarge:
  Arch: HVM64
c4.8xlarge:
  Arch: HVM64
g2.2xlarge:
  Arch: HVMG2
g2.8xlarge:
  Arch: HVMG2
r3.large:
  Arch: HVM64
r3.xlarge:
  Arch: HVM64
r3.2xlarge:
  Arch: HVM64

```
    r3.4xlarge:
      Arch: HVM64
    r3.8xlarge:
      Arch: HVM64
    i2.xlarge:
      Arch: HVM64
    i2.2xlarge:
      Arch: HVM64
    i2.4xlarge:
      Arch: HVM64
    i2.8xlarge:
      Arch: HVM64
    d2.xlarge:
      Arch: HVM64
    d2.2xlarge:
      Arch: HVM64
    d2.4xlarge:
      Arch: HVM64
    d2.8xlarge:
      Arch: HVM64
    hi1.4xlarge:
      Arch: HVM64
    hs1.8xlarge:
      Arch: HVM64
    cr1.8xlarge:
      Arch: HVM64
    cc2.8xlarge:
      Arch: HVM64
  AWSInstanceType2NATArch:
    t1.micro:
      Arch: NATHVM64
    t2.nano:
      Arch: NATHVM64
    t2.micro:
      Arch: NATHVM64
    t2.small:
      Arch: NATHVM64
    t2.medium:
      Arch: NATHVM64
    t2.large:
      Arch: NATHVM64
    m1.small:
```

  Arch: NATHVM64
m1.medium:
  Arch: NATHVM64
m1.large:
  Arch: NATHVM64
m1.xlarge:
  Arch: NATHVM64
m2.xlarge:
  Arch: NATHVM64
m2.2xlarge:
  Arch: NATHVM64
m2.4xlarge:
  Arch: NATHVM64
m3.medium:
  Arch: NATHVM64
m3.large:
  Arch: NATHVM64
m3.xlarge:
  Arch: NATHVM64
m3.2xlarge:
  Arch: NATHVM64
m4.large:
  Arch: NATHVM64
m4.xlarge:
  Arch: NATHVM64
m4.2xlarge:
  Arch: NATHVM64
m4.4xlarge:
  Arch: NATHVM64
m4.10xlarge:
  Arch: NATHVM64
c1.medium:
  Arch: NATHVM64
c1.xlarge:
  Arch: NATHVM64
c3.large:
  Arch: NATHVM64
c3.xlarge:
  Arch: NATHVM64
c3.2xlarge:
  Arch: NATHVM64
c3.4xlarge:

  Arch: NATHVM64
c3.8xlarge:
  Arch: NATHVM64
c4.large:
  Arch: NATHVM64
c4.xlarge:
  Arch: NATHVM64
c4.2xlarge:
  Arch: NATHVM64
c4.4xlarge:
  Arch: NATHVM64
c4.8xlarge:
  Arch: NATHVM64
g2.2xlarge:
  Arch: NATHVMG2
g2.8xlarge:
  Arch: NATHVMG2
r3.large:
  Arch: NATHVM64
r3.xlarge:
  Arch: NATHVM64
r3.2xlarge:
  Arch: NATHVM64
r3.4xlarge:
  Arch: NATHVM64
r3.8xlarge:
  Arch: NATHVM64
i2.xlarge:
  Arch: NATHVM64
i2.2xlarge:
  Arch: NATHVM64
i2.4xlarge:
  Arch: NATHVM64
i2.8xlarge:
  Arch: NATHVM64
d2.xlarge:
  Arch: NATHVM64
d2.2xlarge:
  Arch: NATHVM64
d2.4xlarge:
  Arch: NATHVM64
d2.8xlarge:

```
    Arch: NATHVM64
  hi1.4xlarge:
    Arch: NATHVM64
  hs1.8xlarge:
    Arch: NATHVM64
  cr1.8xlarge:
    Arch: NATHVM64
  cc2.8xlarge:
    Arch: NATHVM64
AWSRegionArch2AMI:
  us-east-1:
    HVM64: ami-0080e4c5bc078760e
    HVMG2: ami-0aeb704d503081ea6
  us-west-2:
    HVM64: ami-01e24be29428c15b2
    HVMG2: ami-0fe84a5b4563d8f27
  us-west-1:
    HVM64: ami-0ec6517f6edbf8044
    HVMG2: ami-0a7fc72dc0e51aa77
  eu-west-1:
    HVM64: ami-08935252a36e25f85
    HVMG2: ami-0d5299b1c6112c3c7
  eu-west-2:
    HVM64: ami-01419b804382064e4
    HVMG2: NOT_SUPPORTED
  eu-west-3:
    HVM64: ami-0dd7e7ed60da8fb83
    HVMG2: NOT_SUPPORTED
  eu-central-1:
    HVM64: ami-0cfbf4f6db41068ac
    HVMG2: ami-0aa1822e3eb913a11
  eu-north-1:
    HVM64: ami-86fe70f8
    HVMG2: ami-32d55b4c
  ap-northeast-1:
    HVM64: ami-00a5245b4816c38e6
    HVMG2: ami-09d0e0e099ecabba2
  ap-northeast-2:
    HVM64: ami-00dc207f8ba6dc919
    HVMG2: NOT_SUPPORTED
  ap-northeast-3:
    HVM64: ami-0b65f69a5c11f3522
```

```
      HVMG2: NOT_SUPPORTED
    ap-southeast-1:
      HVM64: ami-05b3bcf7f311194b3
      HVMG2: ami-0e46ce0d6a87dc979
    ap-southeast-2:
      HVM64: ami-02fd0b06f06d93dfc
      HVMG2: ami-0c0ab057a101d8ff2
    ap-south-1:
      HVM64: ami-0ad42f4f66f6c1cc9
      HVMG2: ami-0244c1d42815af84a
    us-east-2:
      HVM64: ami-0cd3dfa4e37921605
      HVMG2: NOT_SUPPORTED
    ca-central-1:
      HVM64: ami-07423fb63ea0a0930
      HVMG2: NOT_SUPPORTED
    sa-east-1:
      HVM64: ami-05145e0b28ad8e0b2
      HVMG2: NOT_SUPPORTED
    cn-north-1:
      HVM64: ami-053617c9d818c1189
      HVMG2: NOT_SUPPORTED
    cn-northwest-1:
      HVM64: ami-0f7937761741dc640
      HVMG2: NOT_SUPPORTED
Resources:
  EC2Instance:
    Type: 'AWS::EC2::Instance'
    Properties:
      InstanceType: !Ref InstanceType
      SubnetId: !Ref MySubnet
      SecurityGroupIds:
        - !Ref MySG
      KeyName: !Ref KeyName
      ImageId: !FindInMap
        - AWSRegionArch2AMI
        - !Ref 'AWS::Region'
        - !FindInMap
          - AWSInstanceType2Arch
          - !Ref InstanceType
          - Arch
Outputs:
```

```
  InstanceId:
    Description: InstanceId of the newly created EC2 instance
    Value: !Ref EC2Instance
  AZ:
    Description: Availability Zone of the newly created EC2 instance
    Value: !GetAtt
      - EC2Instance
      - AvailabilityZone
  PublicDNS:
    Description: Public DNSName of the newly created EC2 instance
    Value: !GetAtt
      - EC2Instance
      - PublicDnsName
  PublicIP:
    Description: Public IP address of the newly created EC2 instance
    Value: !GetAtt
      - EC2Instance
      - PublicIp
```

Now go to the AWS Console
Select the **N. Virginia** region at the top right of the screen
Click **Services**. Then click **CloudFormation**
Then click **Create Stack**
We can upload the template file or we can create the template in designer.
Select **Create template in Designer**
Then click the **Create template in designer** button
(Here we can actually drag and drop items in the resource types margin on the left onto the canvas in the center and it will automatically generate code for us.
So for example in the left margin click AutosScaling and then drag and drop AutoScalingGroup onto the canvas in the center of the page and you will see code generated at the bottom. Then right click on the resource you created and click the trashcan icon and click delete so that we once again have a blank canvas)
★★★Click the **Template** tab at the bottom of the screen and delete the code that is in there which looks like the following:

```
{
   "AWSTemplateFormatVersion": "2010-09-09",
   "Metadata": {
      "AWS::CloudFormation::Designer": {}
   },
```

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

Then copy all of the YAML code from the template above (the template that you saved in your code editor as templateanatomy.yaml) and paste it into the blank template. You will see that it recognizes the code as YAML because the template language section will be selected.

The screen will say designer is out of date, hit refresh.
Click the **refresh icon** on the top **right** of the page.
Then click the **validate template icon**.
Then click the **create stack icon**.

This will prompt the create stack page. Just click **next**.
This will prompt the specify stack details page.

Stack name: tempanatomyLab ←give it a name

InstanceType: t2.micro ← change it from t2.small to t2.micro
KeyName: ←select a KeyPair that you have on file in the N. Virginia region. Remember that if you need a key pair then you need to create one before you build your stack assuming you don't have any on file that you want to use.

MySG: sg-01b7911e019bf39f7 ←open a new tab and in the aws console in the N. Virginia region go to services and then ec2 and click security groups in the left margin and retrieve the security group id of the security group you would like to use and then copy and paste it into this section

MySubnet: subnet-89ca79c4 ← open a new tab and in the aws console in the N. Virginia region go to **services** and then **vpc** and click **subnets** in the left margin. And for the default n. Virginia vpc, retrieve the subnet ID for us-east 1-1a and copy and paste it into this section. (although we really could select any of the subnets but for simplicity we will select us-east-1a)

SSHLocation: 0.0.0.0/0 ←leave as such (of course a real production environment should be tighter security than this and therefore have a specific ip address range but this is just for demonstration)

Click **next**
This will prompt the configure stack options page.

Tags
Key: name ←give it a key like so
Value: tempanotomy ←give it a value like so

Then click **Next**

This will prompt the review page. (We can actually click on estimate cost to see what it will cost per month)

Scroll to the bottom of the page and click **Create stack**
Wait a minute and then click the **refesh icon** in the Events section and it should say Create_Complete
Click the **Outputs** tab to see the Outputs that we requested in our template which included AZ, instanceID, PublicDNS, and PublicIP

Let's delete our stack now. Click the **Delete** button which is situated above the outputs tab. Then click **Delete stack** to confirm that.

In a few moments the stack should be deleted. If there were an issue it would say delete failed.