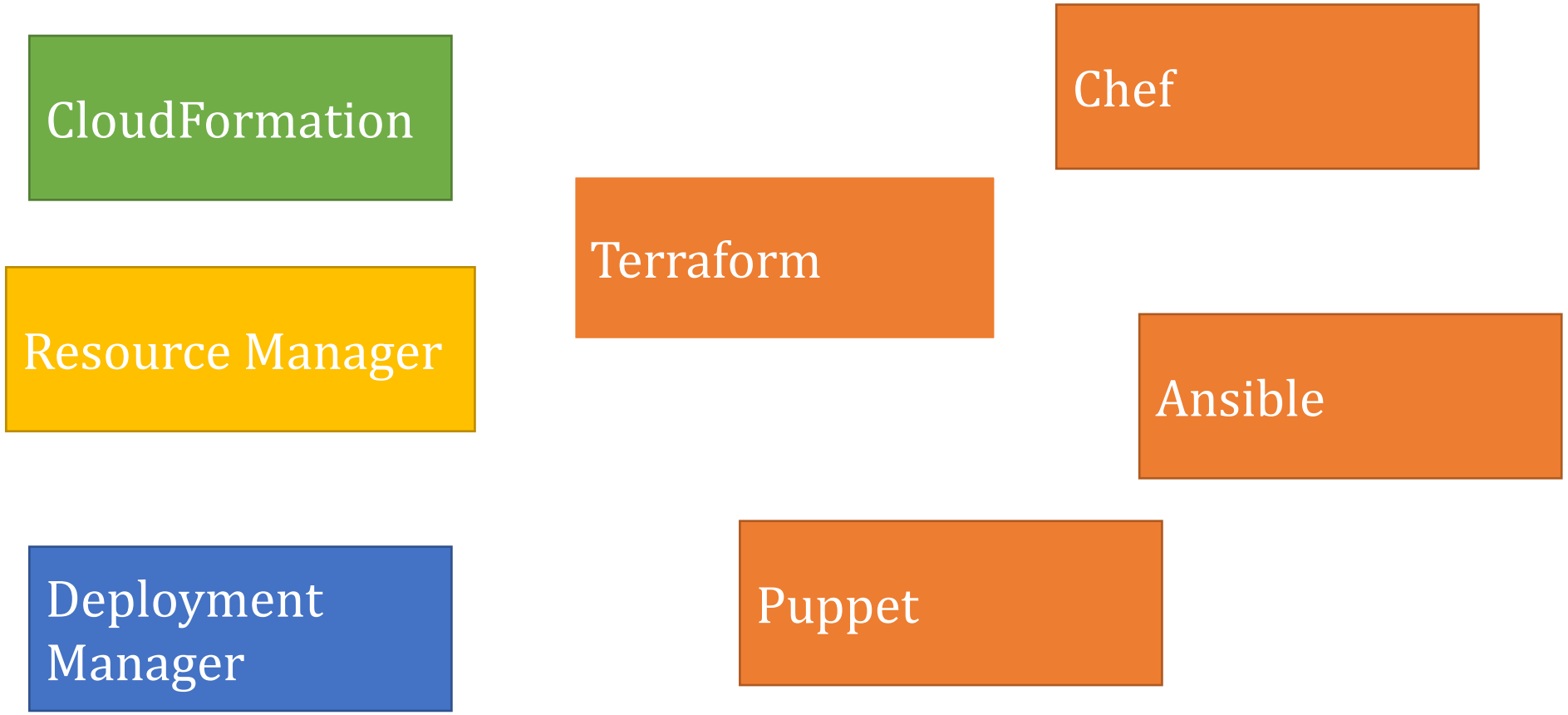
Serverless Data Processing (CSCI 5410)

Dr. Saurabh Dey

Outline

CloudFormation

Services for Cloud Deployment or Infrastructure Automation

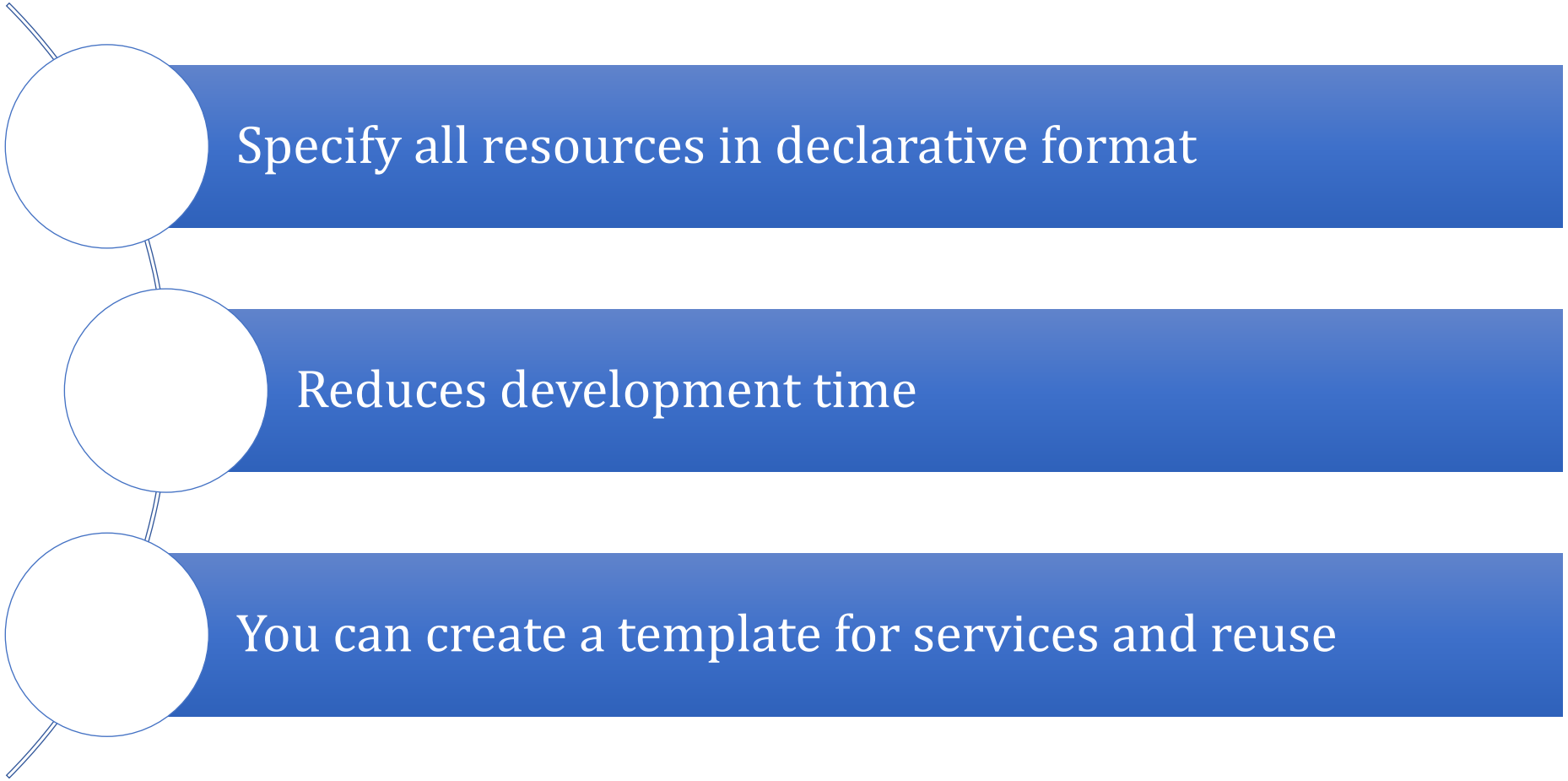


Definitions

- AWS CloudFormation provides a common language for you to model and provision AWS and third party application resources in your cloud environment.
- AWS CloudFormation allows you to use programming languages or a simple text file to model and provision, in an automated and secure manner, all the resources needed for your applications across all regions and accounts.

- Google Cloud Deployment Manager allows you to specify all the resources needed for your application in a declarative format using yaml.
- You can also use Python or Jinja2 templates to parameterize the configuration and allow reuse of common deployment paradigms such as a load balanced, auto-scaled instance group.

Why AWS CloudFormation or GCP Cloud Deployment Manager



AWS CloudFormation Overview

Templates

- An AWS CloudFormation template is a JSON or YAML formatted text file. We can save these files with any extension, such as **.json**, **.yaml**, **.template**, or **.txt**. AWS CloudFormation uses these templates as blueprints for building our AWS resources.

Stacks

- A stack is a collection of AWS resources that you can manage as a single unit. In other words, you can create, update, or delete a collection of resources by creating, updating, or deleting stacks.

```json
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Description" : "A sample template",
  "Resources" : {
    "MyEC2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "ImageId" : "ami-0ff8a91507f77f867",
        "InstanceType" : "t2.micro",
        "KeyName" : "testkey",
        "BlockDeviceMappings" : [
          {
            "DeviceName" : "/dev/sdm",
            "Ebs" : {
              "VolumeType" : "io1",
              "Iops" : "200",
              "DeleteOnTermination" : "false",
              "VolumeSize" : "20"
            }
          }
        ]
      }
    }
  }
}
```

JSON

```yaml
AWSTemplateFormatVersion: "2010-09-09"
Description: A sample template
Resources:
  MyEC2Instance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: "ami-0ff8a91507f77f867"
      InstanceType: t2.micro
      KeyName: testkey
      BlockDeviceMappings:
        -
          DeviceName: /dev/sdm
          Ebs:
            VolumeType: io1
            Iops: 200
            DeleteOnTermination: false
            VolumeSize: 20
```

YAML

```yaml
AWSTemplateFormatVersion: "2010-09-09"
Description: A sample template
Resources:
  MyEC2Instance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: "ami-0ff8a91507f77f867"
      InstanceType: t2.micro
      KeyName: testkey
      BlockDeviceMappings:
        -
          DeviceName: /dev/sdm
          Ebs:
            VolumeType: io1
            Iops: 200
            DeleteOnTermination: false
            VolumeSize: 20
```

```yaml
# Copyright 2016 Google Inc. All rights reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

resources:
- name: vm-created-by-deployment-manager
  type: compute.v1.instance
  properties:
    zone: us-central1-a
    machineType: zones/us-central1-a/machineTypes/n1-standard-1
    disks:
    - deviceName: boot
      type: PERSISTENT
      boot: true
      autoDelete: true
      initializeParams:
        sourceImage: projects/debian-cloud/global/images/family/debian-9
    networkInterfaces:
    - network: global/networks/default
```

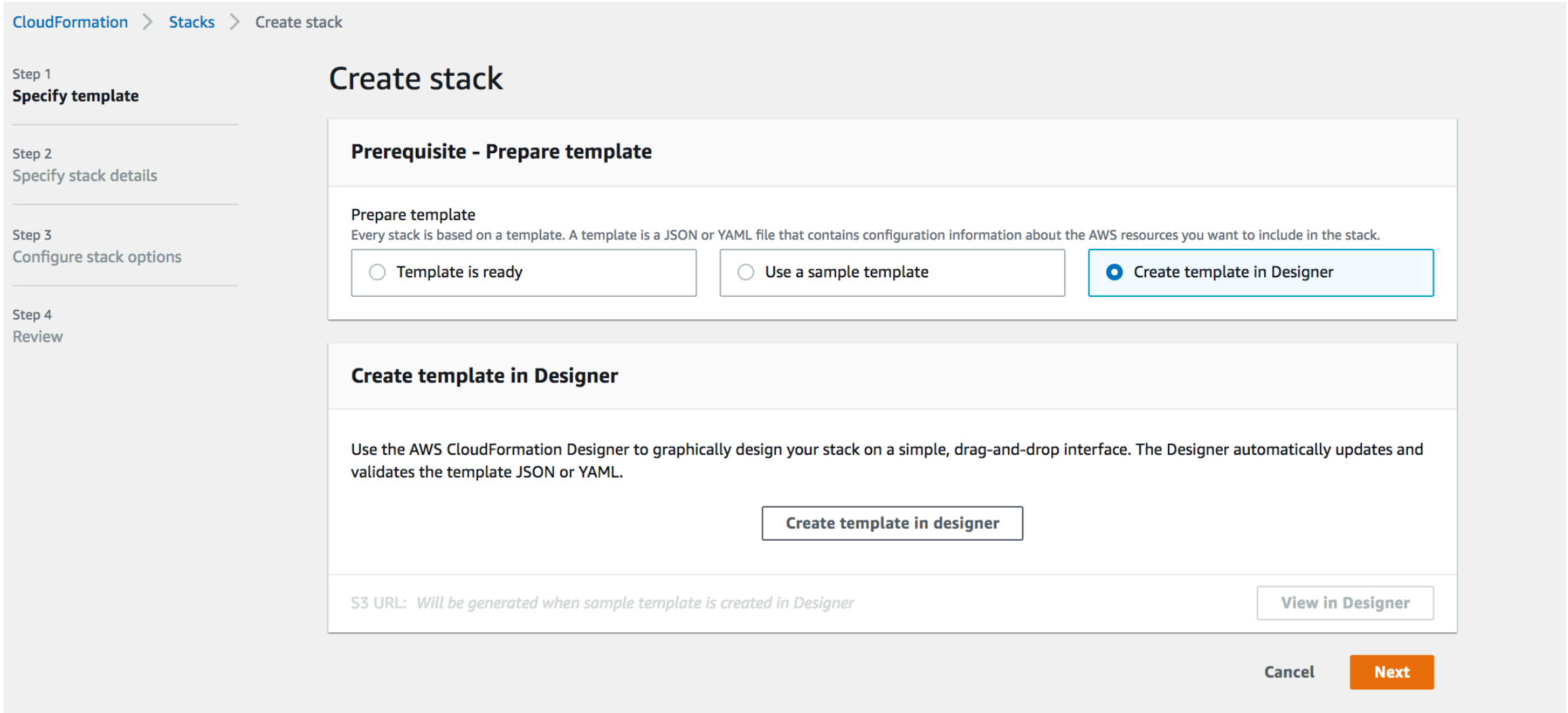
```
{
  "AWSTemplateFormatVersion" : "version date",
  "Description" : "JSON string",
  "Metadata" : {
    template metadata
  },
  "Parameters" : {
    set of parameters
  },
  "Mappings" : {
    set of mappings
  },
  "Conditions" : {
    set of conditions
  },
  "Transform" : {
    set of transforms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

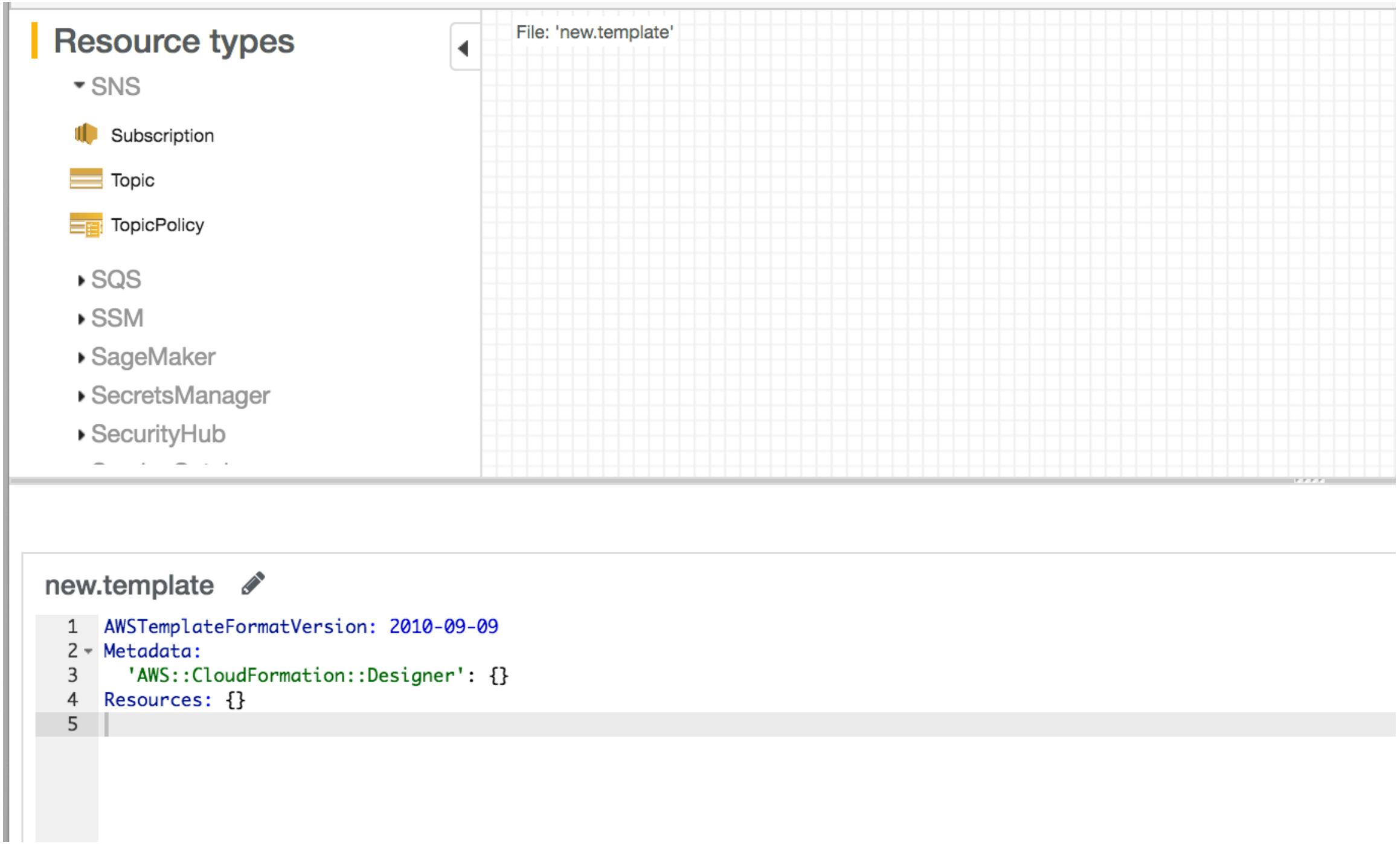
Template Anatomy in JSON

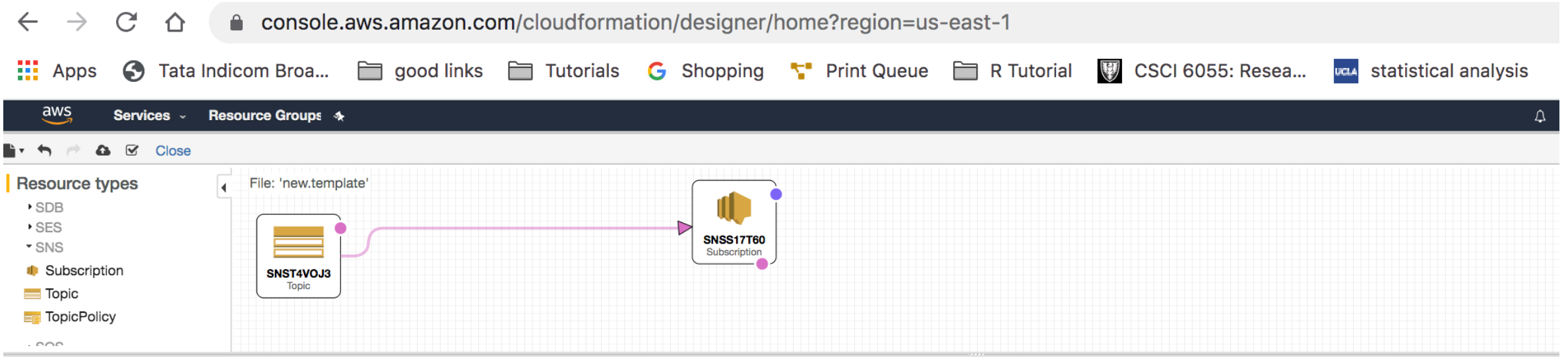
```
---
AWSTemplateFormatVersion: "version date"
Description:
  String
Metadata:
  template metadata
Parameters:
  set of parameters
Mappings:
  set of mappings
Conditions:
  set of conditions
Transform:
  set of transforms
Resources:
  set of resources
Outputs:
  set of outputs
```

Template Anatomy in YAML

Specifies version of AWS SAM


CloudFormation
Stacks
Create stack
Step 1
Specify template
Step 2
Specify stack details
Step 3
Configure stack options
Step 4
Review
Create stack
Prerequisite - Prepare template
Prepare template
Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.
Template is ready
Use a sample template
Create template in Designer
Create template in Designer
Use the AWS CloudFormation Designer to graphically design your stack on a simple, drag-and-drop interface. The Designer automatically updates and validates the template JSON or YAML.
Create template in designer
S3 URL: Will be generated when sample template is created in Designer
View in Designer
Cancel
Next


Resource types
SNS
Subscription
Topic
TopicPolicy
SQS
SSM
SageMaker
SecretsManager
SecurityHub
File: 'new.template'
new.template
1 AWSTemplateFormatVersion: 2010-09-09
2 Metadata:
3 'AWS::CloudFormation::Designer': {}
4 Resources: {}
5



new.template

Choose template language: ○ JSON ● YAML

```
      position:
        x: 401.9506407061396
        'y': 25.619673553965516
      z: 0
    af5cb22d-1198-443b-9c4b-419c93eff48a:
      source:
        id: c36a0a0b-09b8-4d76-b46a-f07e1ecf3ff9
      target:
        id: 0b9ddd8b-c871-4843-b169-0e1e5b22621e
      z: 1
Resources:
  SNST4VOJ3:
    Type: 'AWS::SNS::Topic'
    Properties: {}
    Metadata:
      'AWS::CloudFormation::Designer':
        id: c36a0a0b-09b8-4d76-b46a-f07e1ecf3ff9
    DependsOn:
      - SNSS17T60
  SNSS17T60:
    Type: 'AWS::SNS::Subscription'
    Properties: {}
    Metadata:
      'AWS::CloudFormation::Designer':
        id: 0b9ddd8b-c871-4843-b169-0e1e5b22621e

```

If you want to Deploy Lambda using CloudFormation

```
Resources:
  LambdaZipsBucket:
      Type: AWS::S3::Bucket
  CopyZips:
      Type: Custom::CopyZips
      Properties:
        ServiceToken: !GetAtt 'CopyZipsFunction.Arn'
        DestBucket: !Ref 'LambdaZipsBucket'
        SourceBucket: !Ref 'QSS3BucketName'
        Prefix: !Ref 'QSS3KeyPrefix'
        Objects:
          - functions/packages/MyFunction/lambda.zip
```

https://aws.amazon.com/blogs/infrastructure-and-automation/deploying-aws-lambda-functions-using-aws-cloudformation-the-portable-way/

Questions to Consider

- Can you manage resources outside of CloudFormation?
- What is the difference between CloudFormation and Elastic Beanstalk?

Thank You