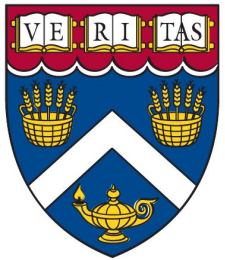


HARVARD
Extension School

Week 7
AWS Services - DevOps Tools

This week..

- **Week 07:** AWS Services - DevOps Tools

This lecture will discuss some of the DevOps resources, tools, and methods introduced by AWS like CloudFormation, Auto Scaling, CloudWatch, CloudTrail and Cloud Config. The focus will be on the use of Auto-scaling for high availability and scalability, and on CloudFormation to build AWS resources using CloudFormation templates.

- High availability and scalability
  - Auto Scaling
  - Elastic Load Balancer
- CloudFormation
  - Templates
  - Template Designer
  - Templates output and exports
- Monitoring and Logging
  - CloudWatch
  - CloudTrail
  - CloudConfig
  - Elasticsearch

Assignment 3

DevOps & Agile - reminder

- DevOps values:
  - Agility: Do things faster
  - Cost Efficiency: Do things cheaper
  - Quality: Do things that will meet the needs of the business and end users.

Keys to consider

When choosing devops tools:

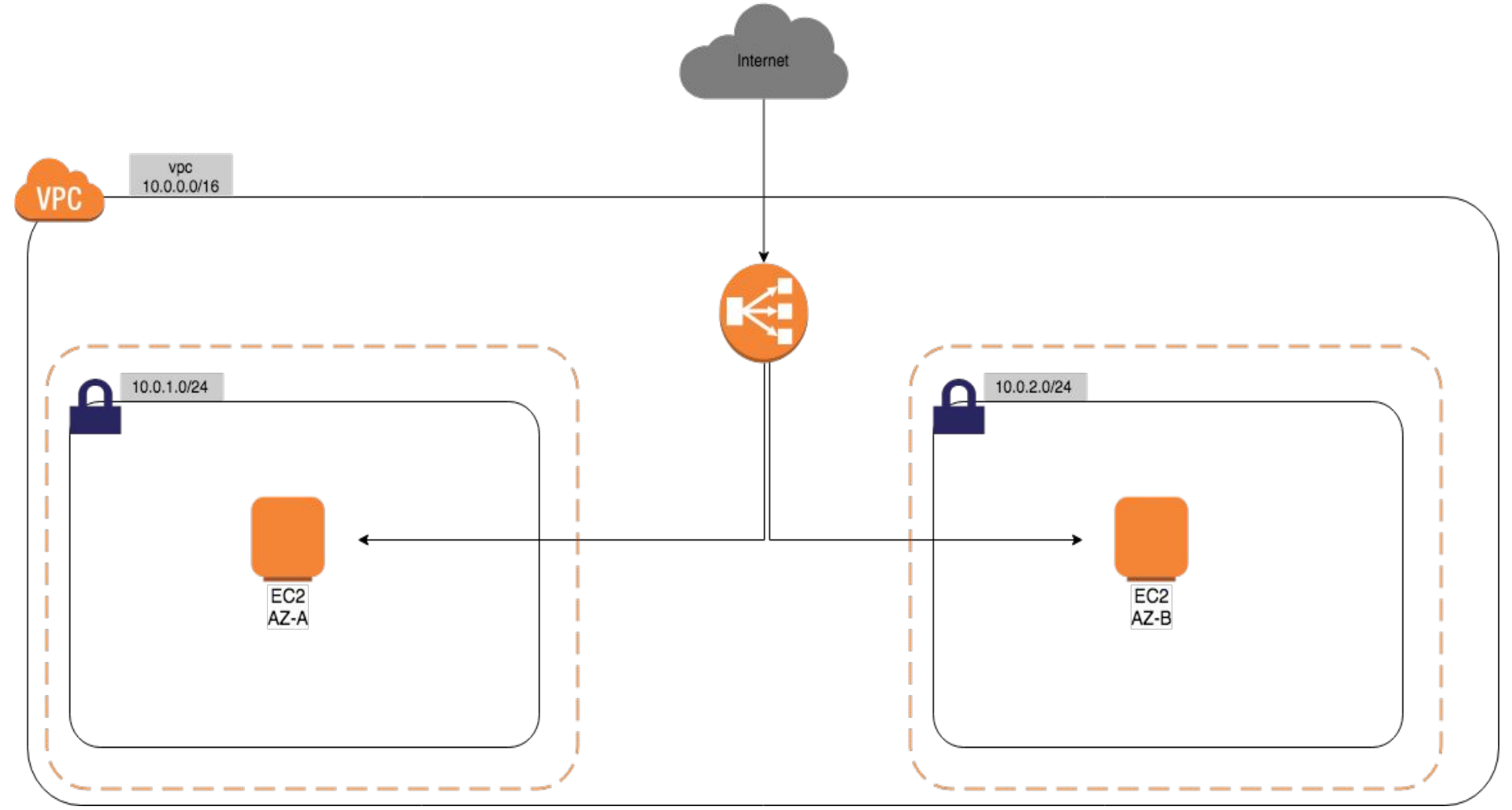
- Automate as much as possible
- Cost
- Integration with the cloud platform in use
- Developers acceptance
- Tools will be replaceable
- Not limited to services/tools provided by the cloud platform in use
- It’s ok to pick the wrong tool but don’t continue using the wrong tool

Testing tools process

1. Test the chosen tool as a unit
2. Test the chosen tool with other tools in place
3. Test the complete process

EC2 Load Balancers for availability

- Service based on software to functionally route a changing number of EC2 instances in different AZs within the same region.
- The Load Balancer is a single point of contact to the outside.
- Two types of Load Balancers:
  - Monitor health check at the application level (ALB - Application Load Balancers): preferred for http or https
  - Monitor at the network level (ELB - Elastic Load Balancers): preferred for http or https as well as TCP or SSL
- If any of the EC2s in a specific Target Group in any of the zones become unavailable, the Load Balancer simply re-routes traffic to the available servers.


Internet
VPC
vpc
10.0.0.0/16
10.0.1.0/24
EC2
AZ-A
10.0.2.0/24
EC2
AZ-B

Load Balancer

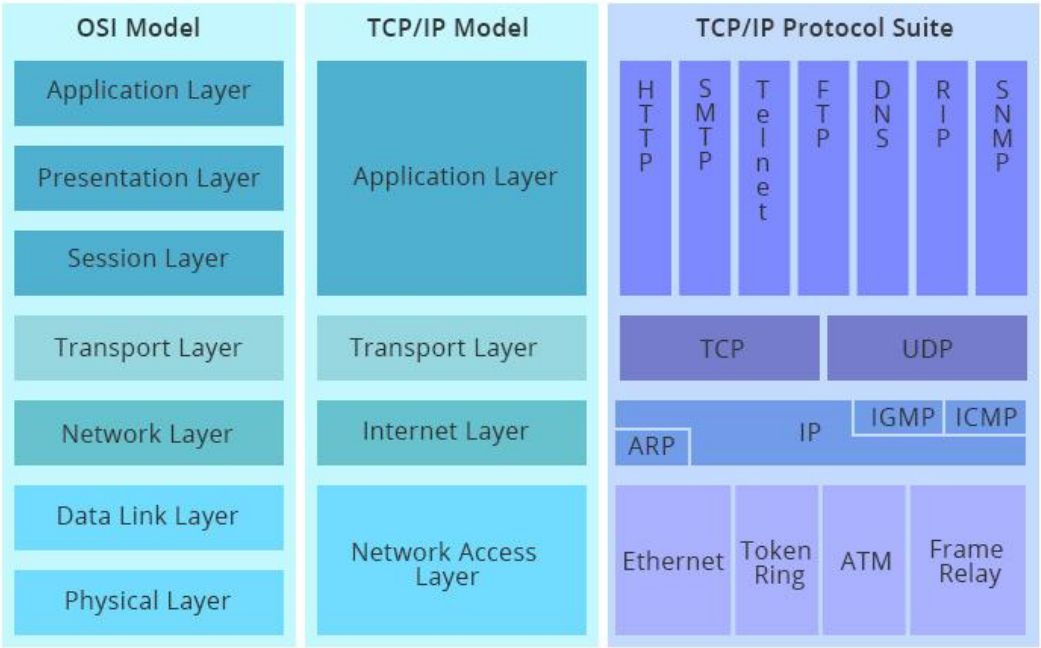
- ➔ Layer 7 HTTP/HTTPS Application Layer Load Balancer
  - ◆ Target groups and Rules to route paths

- ➔ Layer 4 Network Load Balancer
  - ◆ Extreme performance is required.
  - ◆ Low latency while handling millions of requests per second

- ➔ Classic Load Balancer: Layer 4 and can use layer 7
  - ◆ X-Forwarding
  - ◆ sticky session

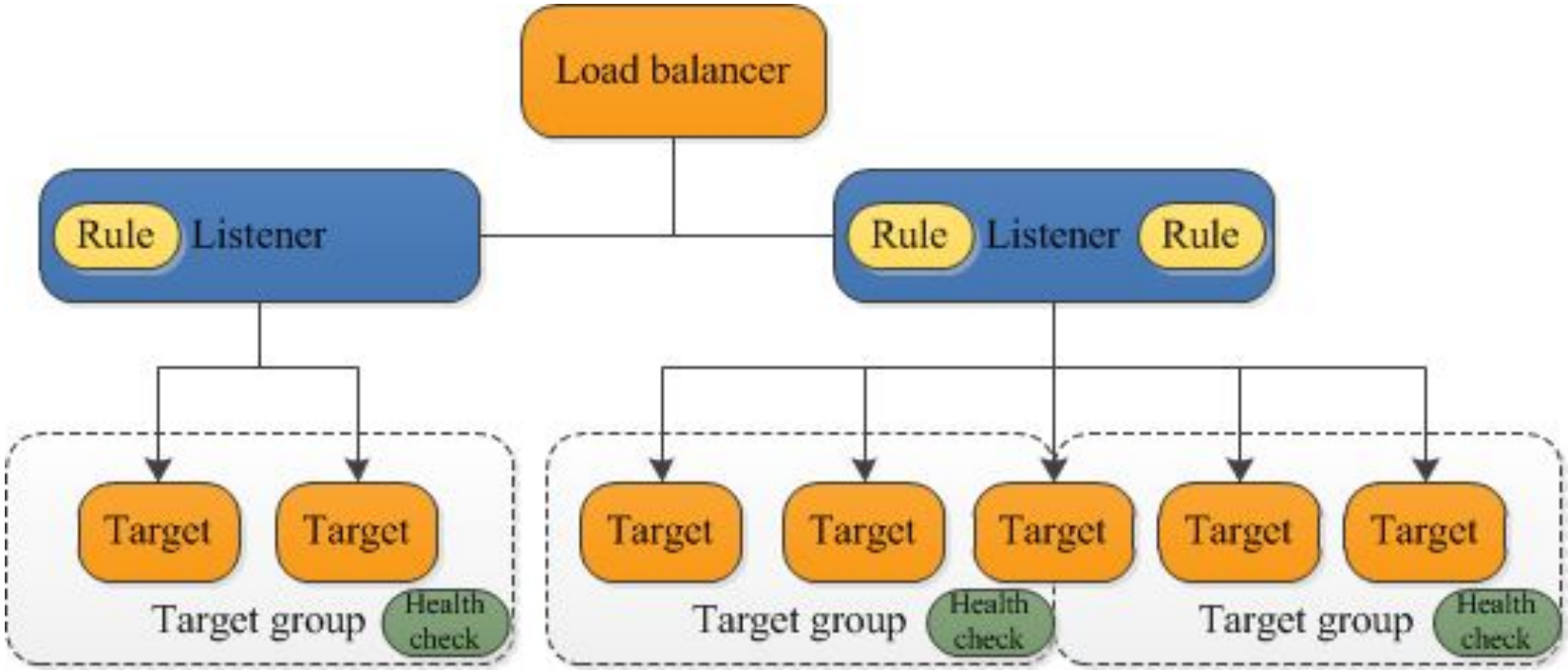
https://aws.amazon.com/elasticloadbalancing/features/

TCPI/IP and OSI model



https://community.fs.com/blog/tcpip-vs-osi-whats-the-difference-between-the-two-models.html

Application Load Balancer



https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html

Network Load Balancer

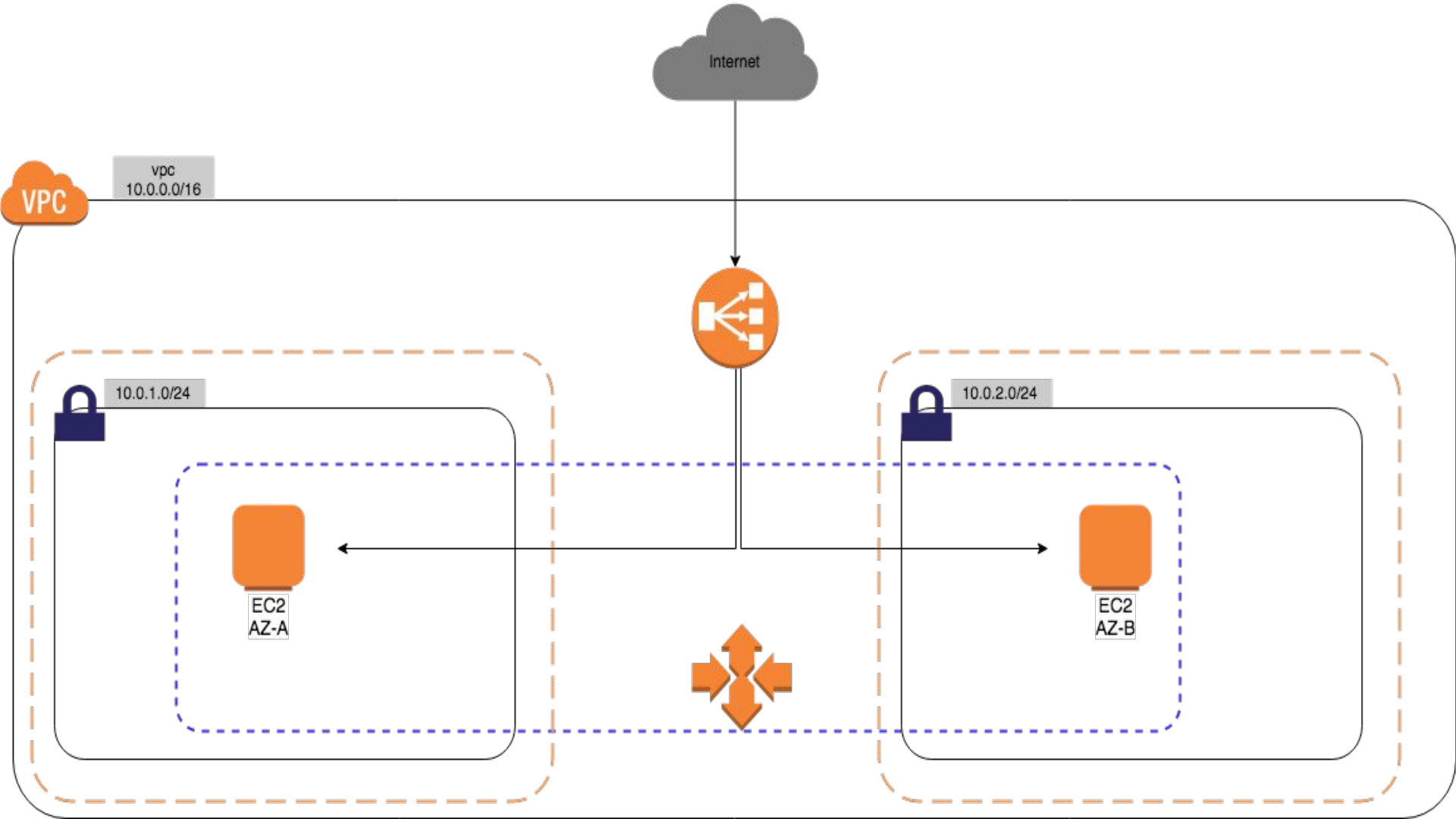
https://docs.aws.amazon.com/elasticloadbalancing/latest/network/introduction.html

Configure AWS Load Balancer

1. Create a Target Group
    a. Protocol
        i. TCP
        ii. HTTP
        iii. HTTPS
    b. Target
        i. Instance
        ii. IP
2. Register instances(CLB)
3. Create ALB or NLB
4. Select the VPC and AZs
5. Configure security groups
6. Select the Target Group created earlier
7. Register targets

Auto Scaling groups for availability

- Components are:
  - EC2 instances are organized into groups for the purpose of scaling and management.
  - EC2 auto scaling group uses a launch configuration as a template for the instances.
  - Scaling plan: to tell Auto Scaling group when and how to scale.


Internet
VPC
vpc
10.0.0.0/16
10.0.1.0/24
10.0.2.0/24
EC2
AZ-A
EC2
AZ-B

Configure AWS Auto Scaling

From the console:

1. Create an Auto Scaling Group
2. Create launch configuration: the template used to launch auto scaling instances
    a. Select a pre-configured AMI
    b. Select type of instance
    c. Attach IAM role (Best Practice)
    d. Add storage if needed
3. Configure the auto scaling group details
    a. Define the group size
    b. Select the VPC
    c. Associate Subnets
    d. Use auto scaling group initial size or use policy
    e. Configure Notification (optional)

Update Launch Configuration

To update launch configuration :

1) Copy the launch configuration and choose new name
    a) Update the user data
    b) Make sure all config is ok
2) Replace the Autoscaling with the new Launch Configuration
3) Remove instances: for each instance in the LB
    a) terminate the instance
    b) wait for the new instance to come up.
    c) Test the new instance functionality
4) Remove the old launch configuration

CloudFormation

- Infrastructure as code: to build an automated delivery process for infrastructure, and updating it as needed.
- Express AWS resources in a text file in either a JSON format or YAML format.
- All template building blocks are optional except for “Resources”.
- Each template create a cloudformation stack

Cloudformation Template blocks

```
{
  "AWSTemplateFormatVersion" : "version date",
  "Description" : "JSON string",
  "Metadata" : {
    template metadata
e.g:
    "Instances" : {"Description" : "Information about the instances"}

  },
  "Parameters" : {
    set of parameters
e.g.
    "InstanceType" : {
        "Type" : "String",
        "Default" : "m1.small",
        "Description" : "EC2 instance type, e.g. m1.small, m1.large, etc."
   }
```

The latest template format version is 2010-09-09

String between 0 and 1024 bytes

Implementation details about specific resources

Create custom values as input when creating a stack

```
  "Mappings" : {
    set of mappings
e.g.
 "RegionMap" : {
    "us-east-1"      : { "HVM64" : "ami-0ff8a91507f77f867"}
  },
  "Conditions" : {
    set of conditions
  },
  "Resources" : {
    set of resources
e.g.
  "MyEC2Instance" : {
    "Type" : "AWS::EC2::Instance",
    "Properties" : {
      "ImageId" : "ami-0ff8a91507f77f867"
    }
  },
  "Outputs" : {
    set of outputs
e.g.
"InstanceID" : {
    "Description": "The Instance ID",
    "Value" : { "Ref" : "EC2Instance" }
  }
}
```

Matches a key to a corresponding set of named values

Used to match parameter values with resources or properties if the condition is true.

Specifies the stack resources and their properties

Declares output values that you can import into other stacks

Benefits of Cloudformation

- Automatically create and delete resources in order
- Update resources by updating the template
- One can monitor events that take place while cloudformation stack building is taking place from the console Events tab.
- Protection from deleting resources is available
- Specify deletion policies in template for specific resources
- If anything goes wrong while creating, deleting, and updating Cloudformation will rollback to the previous state automatically

Create VPC Environment

```
AWSTemplateFormatVersion: 2010-09-09

Description: "AWS CloudFormation Template to create a VPC with one public subnet and a private on"

Mappings:
  CIDRConfig:
    VPC:
      CIDR: 10.10.0.0/16
    Public:
      CIDR: 10.10.0.0/24
    Private:
      CIDR: 10.10.128.0/17

Resources:
  VPC:
    Type: 'AWS::EC2::VPC'
    Properties:
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
      CidrBlock: !FindInMap
        - CIDRConfig
        - VPC
        - CIDR
      Tags:
        - Key: Name
          Value: !Sub '${AWS::StackName}-vpc'
```

```
PublicSubnet:
  Type: 'AWS::EC2::Subnet'
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !FindInMap
      - CIDRConfig
      - Public
      - CIDR
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-public'

PrivateSubnet:
  Type: 'AWS::EC2::Subnet'
  Properties:
    VpcId: !Ref VPC
    CidrBlock: !FindInMap
      - CIDRConfig
      - Private
      - CIDR
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-private'
```

```
InternetGateway:
  Type: 'AWS::EC2::InternetGateway'
  Properties:
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-IGW'

GatewayToInternet:
  Type: 'AWS::EC2::VPCGatewayAttachment'
  Properties:
    VpcId: !Ref VPC
    InternetGatewayId: !Ref InternetGateway
```

```
NAT:
  Type: 'AWS::EC2::NatGateway'
  Properties:
    AllocationId: !GetAtt
      - EIP
      - AllocationId
    SubnetId: !Ref PublicSubnet
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-NAT'
EIP:
  Type: 'AWS::EC2::EIP'
  Properties:
    Domain: vpc
```

```
PublicRouteTable:
  Type: 'AWS::EC2::RouteTable'
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-publicRT'

PublicRoute:
  Type: 'AWS::EC2::Route'
  DependsOn: InternetGateway
  Properties:
    RouteTableId: !Ref PublicRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    GatewayId: !Ref InternetGateway

PublicSubnetRouteTableAssociation:
  Type: 'AWS::EC2::SubnetRouteTableAssociation'
  Properties:
    SubnetId: !Ref PublicSubnet
    RouteTableId: !Ref PublicRouteTable
```

```
PrivateRouteTable:
  Type: 'AWS::EC2::RouteTable'
  Properties:
    VpcId: !Ref VPC
    Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-privateRT'

PrivateRoute:
  Type: 'AWS::EC2::Route'
  DependsOn: NAT
  Properties:
    RouteTableId: !Ref PrivateRouteTable
    DestinationCidrBlock: 0.0.0.0/0
    NatGatewayId: !Ref NAT

PrivateSubnetRouteTableAssociation:
  Type: 'AWS::EC2::SubnetRouteTableAssociation'
  Properties:
    SubnetId: !Ref PrivateSubnet
    RouteTableId: !Ref PrivateRouteTable
```

```
Outputs:

  VPCId:
    Description: VPCId of the newly created VPC
    Value: !Ref VPC
    Export:
      Name: !Sub '${AWS::StackName}-VPCID'

  PublicSubnet:
    Description: SubnetId of the public subnet
    Value: !Ref PublicSubnet
    Export:
      Name: !Sub '${AWS::StackName}-PublicSubID'

  PrivateSubnet:
    Description: SubnetId of the private subnet
    Value: !Ref PrivateSubnet
    Export:
      Name: !Sub '${AWS::StackName}-PrivateSubID'
```

Create EC2 Instance

```
AWSTemplateFormatVersion: 2010-09-09
Description: "AWS CloudFormation Template EC2InstanceWithSecurityGroup: Create an Amazon EC2"
Parameters:
  UserName:
    Type: String
    Default: e91
  SSHPublicKey:
    Type: String
    Default: ssh-rsa AAAAB3NzaC1yc2EAAAADAQABA.........
  InstanceType:
    Description: EC2 instance type
    Type: String
    Default: t2.micro
    AllowedValues:
      - t2.micro
      - t2.small
      - t2.medium
    ConstraintDescription: must be a valid EC2 instance type.
  AMI:
    Description: AMI to create the EC2
    Type: String
    Default: ami-0dc7769bc7e889a35
  TagName:
    Description: Value of the Name tag for this EC2
    Type: String
    Default: e91
```

```
Resources:
  SSHANDWEB:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupName: SSH_LOCALWEB_INTERNET
      GroupDescription: ssh access to my machine and 80 and 443 to the internet
      VpcId: !ImportValue
        'Fn::Sub': myStack1-VPCID
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: '80'
          ToPort: '80'
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: '443'
          ToPort: '443'
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: '22'
          ToPort: '22'
          CidrIp: 140.247.0.0/16

  nwFace1:
    Type: 'AWS::EC2::NetworkInterface'
    Properties:
      SubnetId: !ImportValue
```

```
EC2Instance:
  Type: 'AWS::EC2::Instance'
  Properties:
    InstanceType: !Ref InstanceType
    NetworkInterfaces:
      - NetworkInterfaceId: !Ref nwFace1
        DeviceIndex: '0'
    ImageId: !Ref AMI
    Tags:
      -
        Key: Name
        Value: !Ref TagName
    UserData:
      Fn::Base64:
        Fn::Sub:
          - |
            #!/bin/bash -xe
            adduser ${USERNAME}
            echo ${USERNAME} 'ALL=(ALL) NOPASSWD:ALL'>>/etc/sudoers.d/${USERNAME}
            mkdir /home/${USERNAME}/.ssh
            echo ${SSHKEY} > /home/${USERNAME}/.ssh/authorized_keys
            chown -R ${USERNAME}.${USERNAME} /home/${USERNAME}/.ssh
            chmod 700 /home/${USERNAME}/.ssh
            chmod 600 /home/${USERNAME}/.ssh/authorized_keys
            hostnamectl set-hostname  mystack1.harvard.edu --pretty
            hostnamectl set-hostname  mystack1.harvard.edu --static
```

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description" : "This template creates a web server. **WARNING** This template creates an Amazon EC2
instance. You will be billed for the AWS resources used if you create a stack from this template.",

  "Parameters" : {

    "KeyName": {
      "Description" : "Name of an existing EC2 KeyPair to enable SSH access to the instances",
      "Type": "AWS::EC2::KeyPair::KeyName",
      "ConstraintDescription" : "must be the name of an existing EC2 KeyPair."
    },

    "InstanceType" : {
      "Description" : "WebServer EC2 instance type",
      "Type" : "String",
      "Default" : "t1.micro",
      "AllowedValues" : [ "t1.micro", "m3.medium", "m3.large", "c3.large", "c3.xlarge" ],
      "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",
```

Monitoring and Logging

- What to monitor?
  - Availability - functional?
  - Security -
  - Performance - fast enough?
  -

Resources

Application Load Balancer:
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html

AS HealthCheck: https://docs.aws.amazon.com/autoscaling/ec2/userguide/healthcheck.html

ELB features: https://aws.amazon.com/elasticloadbalancing/features/