

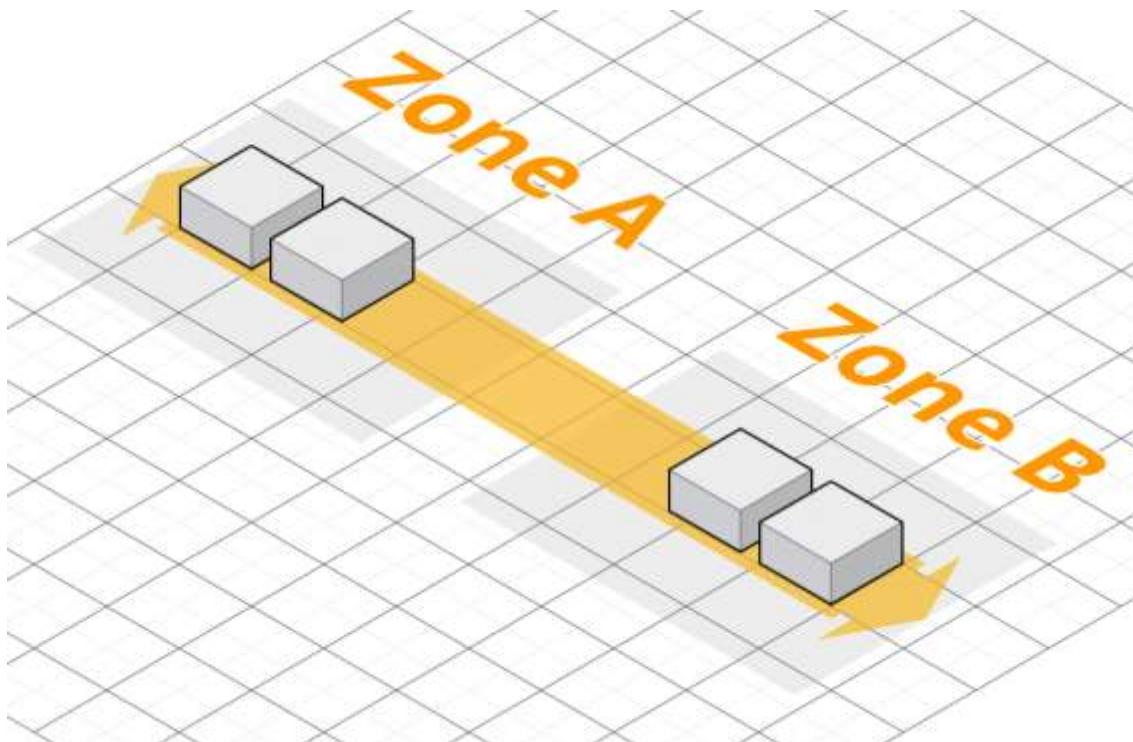
Praveen kumar H A 7204866296

## Rolling Update with AWS CloudFormation

Updating all EC2 instances in an Auto Scaling Group becomes complex and expensive when done manually. If you follow the immutable infrastructure approach you will roll out a new version by creating new EC2 instances. You never change the old instances. But how do you deploy security patches of the OS or a new version of your application automatically? One option is using AWS CloudFormation.

CloudFormation supports the `UpdatePolicy` attribute for Auto Scaling Groups. You can define that CloudFormation performs a [rolling update](#) of instances whenever the Launch Configuration changes. A rolling update will roll out your change in small batches.

Here is an example. Let's say we have 4 EC2 instances running version A (grey) and a batch size of 2. Now we roll out version B (dark). Have a look at the following animation.




1. Two EC2 instances with version B are started (blue).
2. Wait until both new EC2 instances running version B are in service (dark). You now have 6 EC2 instances running.
3. Terminate the two oldest EC2 instances running version A. You now have 4 EC2 instances running.

4. Two EC2 instances with version B are started (blue).
5. Wait until both new EC2 instances running version B are in service (dark).  
You now have 6 EC2 instances running.
6. Terminate the two oldest EC2 instances running version A. You now have 4 EC2 instances running.
7. Done.

The following CloudFormation template is enriched with some vocal comments and line highlighting. Press the play button and enjoy the show.

How do you like CloudFormation templates annotated with voice? [Let me know!](#)

Try rolling updates with our CloudFormation template:

1. 
2. Click **Next** to proceed with the next step of the wizard.
3. Specify a name and all parameters for the stack. Set **VersionParameter** to 1.
4. Click **Next** to proceed with the next step of the wizard.
5. Click **Next** to skip the **Options** step of the wizard.
6. Click **Create** to start the creation of the stack.
7. Wait until the stack reaches the state `CREATE_COMPLETE`.
8. Open the **Outputs** tab and click on the provided ELB URL. You should see a white page that contains *Version 1* text.
9. In the CloudFormation console, select the stack, click the **Actions** button at the top, select **Update Stack**.
10. Click **Next** to proceed with the next step of the wizard.
11. Change the **VersionParameter** to 2.
12. Click **Next** to proceed with the next step of the wizard.
13. Click **Next** to skip the **Options** step of the wizard.
14. Click **Update** to update the stack.
15. Go back to the page (ELB URL) and reload every now and then. After a few minutes it should switch to *Version 2*.
16. Don't forget to delete the CloudFormation stack!

Here it is sample cloud formation template you can refer

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "Rolling Update",
  "Parameters": {
    "VPCParameter": {
      "Description": "VPC to launch instances into",
```

```
"Type": "AWS::EC2::VPC::Id"

},

"SubnetsParameter": {

  "Description": "Subnets to launch instances into",

  "Type": "List<AWS::EC2::Subnet::Id>"

},

"VersionParameter": {

  "Description": "Software version to deploy",

  "Type": "String"

}

},

"Mappings": {

  "RegionMap": {

    "eu-west-1": {"AMI": "ami-bff32ccc"},

    "ap-southeast-1": {"AMI": "ami-c9b572aa"},

    "ap-southeast-2": {"AMI": "ami-48d38c2b"},

    "eu-central-1": {"AMI": "ami-bc5b48d0"},

    "ap-northeast-2": {"AMI": "ami-249b554a"},

    "ap-northeast-1": {"AMI": "ami-383c1956"},

    "us-east-1": {"AMI": "ami-60b6c60a"},

    "sa-east-1": {"AMI": "ami-6817af04"},

    "us-west-1": {"AMI": "ami-d5ea86b5"},

    "us-west-2": {"AMI": "ami-f0091d91"}

  }

},
```

```
"Resources": {  
  "ELBSecurityGroup": {  
    "Type": "AWS::EC2::SecurityGroup",  
    "Properties": {  
      "GroupDescription": "elb-sg",  
      "SecurityGroupEgress": [{  
        "IpProtocol": "-1",  
        "CidrIp": "0.0.0.0/0"  
      }],  
      "SecurityGroupIngress": [{  
        "CidrIp": "0.0.0.0/0",  
        "FromPort": 80,  
        "IpProtocol": "tcp",  
        "ToPort": 80  
      }],  
      "VpcId": {"Ref": "VPCParameter"}  
    }  
  },  
  "ELB": {  
    "Type": "AWS::ElasticLoadBalancing::LoadBalancer",  
    "Properties": {  
      "ConnectionDrainingPolicy": {  
        "Enabled": true,  
        "Timeout": 30  
      },  
    },  
  },  
}
```

```
"CrossZone": true,

"HealthCheck": {

  "HealthyThreshold": "2",

  "Interval": "10",

  "Target": "HTTP:80/",

  "Timeout": "5",

  "UnhealthyThreshold": "2"

},

"LoadBalancerName": "elb",

"Listeners": [{

  "InstancePort": "80",

  "InstanceProtocol": "HTTP",

  "LoadBalancerPort": "80",

  "Protocol": "HTTP"

}],

"Scheme": "internet-facing",

"SecurityGroups": [{"Ref": "ELBSecurityGroup"}],

"Subnets": {"Ref": "SubnetsParameter"}

}

},

"EC2SecurityGroup": {

  "Type": "AWS::EC2::SecurityGroup",

  "Properties": {

    "GroupDescription": "ec2-sg",

    "SecurityGroupEgress": [{
```

```

        "IpProtocol": "-1",
        "CidrIp": "0.0.0.0/0"
    }],
    "SecurityGroupIngress": [{
        "CidrIp": "0.0.0.0/0",
        "FromPort": 22,
        "IpProtocol": "tcp",
        "ToPort": 22
    }, {
        "FromPort": 80,
        "IpProtocol": "tcp",
        "SourceSecurityGroupId": {"Ref": "ELBSecurityGroup"},
        "ToPort": 80
    }],
    "VpcId": {"Ref": "VPCParameter"}
}
},
"LaunchConfiguration": {
    "Type": "AWS::AutoScaling::LaunchConfiguration",
    "Properties": {
        "AssociatePublicIpAddress": true,
        "ImageId": {"Fn::FindInMap": ["RegionMap", {"Ref": "AWS::Region"}, "AMI"]},
        "InstanceType": "t2.micro",
        "SecurityGroups": [{"Ref": "EC2SecurityGroup"}],
        "UserData": {"Fn::Base64": {"Fn::Join": ["", [

```

```

        "#!/bin/bash -ex", "\n",

        "yum -y install httpd", "\n",

        "chkconfig httpd on", "\n",

        "echo \"Version ", {"Ref": "VersionParameter"}, "\"" >>
/var/www/html/index.html", "\n",

        "service httpd start", "\n",

        "/opt/aws/bin/cfn-signal -e 0 --region ", {"Ref": "AWS::Region"}, " --stack ",
{"Ref": "AWS::StackName"}, " --resource AutoScalingGroup", "\n"

    ]]]}

}

},

"AutoScalingGroup": {

    "Type": "AWS::AutoScaling::AutoScalingGroup",

    "Properties": {

        "MinSize": "1",

        "MaxSize": "2",

        "DesiredCapacity": "1",

        "LaunchConfigurationName": {"Ref": "LaunchConfiguration"},

        "VPCZoneIdentifier": {"Ref": "SubnetsParameter"},

        "LoadBalancerNames": [{"Ref": "ELB"}],

        "HealthCheckGracePeriod": 60,

        "HealthCheckType": "ELB"

    },

    "CreationPolicy": {

        "ResourceSignal": {

            "Count": 1,

```

```
    "Timeout": "PT10M"
  }
},
"UpdatePolicy": {
  "AutoScalingRollingUpdate": {
    "MaxBatchSize": "2",
    "MinInstancesInService": "1",
    "PauseTime": "PT10M",
    "SuspendProcesses": ["AlarmNotification"],
    "WaitOnResourceSignals": true
  }
}
},
"Outputs": {
  "ELB": {
    "Value": {"Fn::Join": ["", ["http://", {"Fn::GetAtt": ["ELB", "DNSName"]}]]},
    "Description": "Load Balancer URL"
  }
}
}
```