How to Jfrog artifactory HA in AWS ECS using cloudformation

Prerequisites:

1. AWS ECS AND EKS (CLOUDFORMATION)

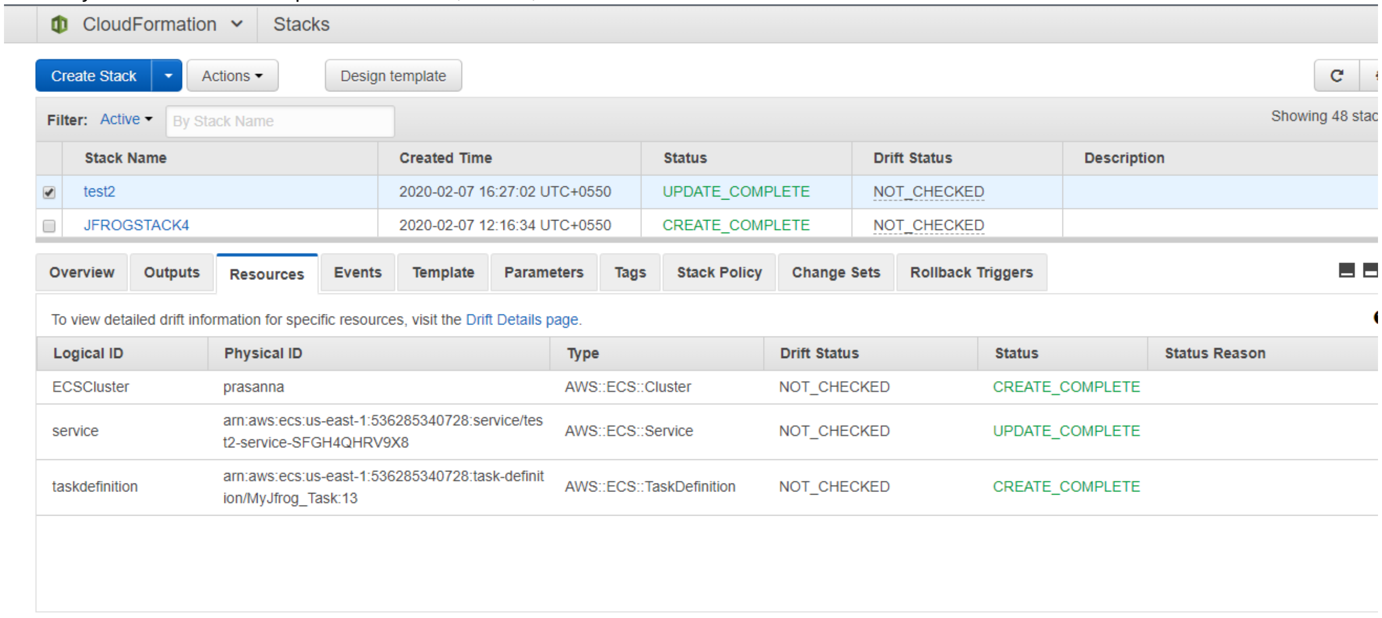
When we say high availability, we are referring to systems that can operate continuously without failure for a long time. The term implies that the system has been tested thoroughly to stand any sort of failure. Jenkins is a crucial component of DevOps and its downtime may have adverse effects on the DevOps environment. To overcome these, we need a high availability setup for Jfrog artifactory.

Go to VS code IDE and edit and push the code to aws cloudformation as below.

aws cloudformation create-stack --template-body file://test2.yaml --stack-name test2.

Go to Cloudformation and check the stack.

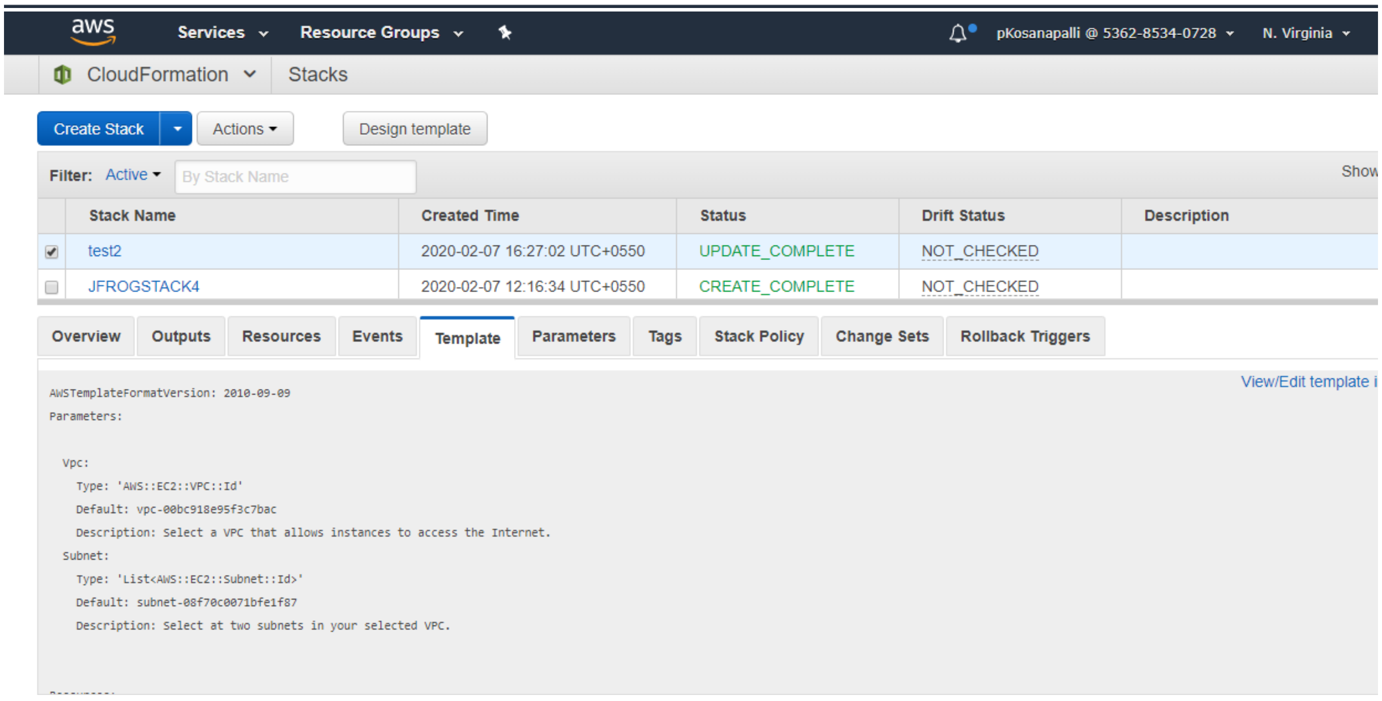
1. you can check the template of the stack, events, resources that created for this stack and other details as below.



	Stack Name	Created Time	Status	Drift Status	Description
✓	test2	2020-02-07 16:27:02 UTC+0550	UPDATE_COMPLETE	NOT_CHECKED	
	JFROGSTACK4	2020-02-07 12:16:34 UTC+0550	CREATE_COMPLETE	NOT_CHECKED	

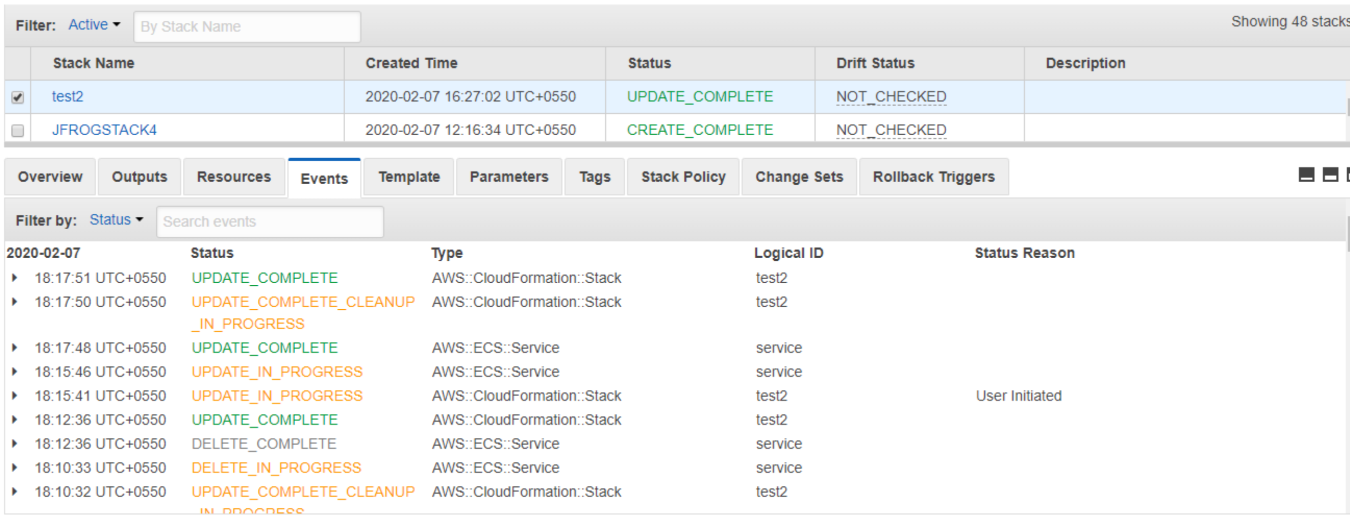
To view detailed drift information for specific resources, visit the Drift Details page.

Logical ID	Physical ID	Type	Drift Status	Status	Status Reason
ECSCluster	prasanna	AWS::ECS::Cluster	NOT_CHECKED	CREATE_COMPLETE	
service	arn:aws:ecs:us-east-1:536285340728:service/test2-service-SFGH4QHRV9X8	AWS::ECS::Service	NOT_CHECKED	UPDATE_COMPLETE	
taskdefinition	arn:aws:ecs:us-east-1:536285340728:task-definition/MyJfrog_Task:13	AWS::ECS::TaskDefinition	NOT_CHECKED	CREATE_COMPLETE	



	Stack Name	Created Time	Status	Drift Status	Description
✓	test2	2020-02-07 16:27:02 UTC+0550	UPDATE_COMPLETE	NOT_CHECKED	
	JFROGSTACK4	2020-02-07 12:16:34 UTC+0550	CREATE_COMPLETE	NOT_CHECKED	

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:

  Vpc:
    Type: 'AWS::EC2::VPC::Id'
    Default: vpc-00bc918e95f3c7bac
    Description: Select a VPC that allows instances to access the Internet.
  Subnet:
    Type: 'List<AWS::EC2::Subnet::Id>'
    Default: subnet-08f70c0071bfe1f87
    Description: Select at two subnets in your selected VPC.
```



Filter: Active | By Stack Name | Showing 48 stacks

	Stack Name	Created Time	Status	Drift Status	Description
✓	test2	2020-02-07 16:27:02 UTC+0550	UPDATE_COMPLETE	NOT_CHECKED	
	JFROGSTACK4	2020-02-07 12:16:34 UTC+0550	CREATE_COMPLETE	NOT_CHECKED	

2020-02-07	Status	Type	Logical ID	Status Reason
18:17:51 UTC+0550	UPDATE_COMPLETE	AWS::CloudFormation::Stack	test2	
18:17:50 UTC+0550	UPDATE_COMPLETE_CLEANUP_IN_PROGRESS	AWS::CloudFormation::Stack	test2	
18:17:48 UTC+0550	UPDATE_COMPLETE	AWS::ECS::Service	service	
18:15:46 UTC+0550	UPDATE_IN_PROGRESS	AWS::ECS::Service	service	
18:15:41 UTC+0550	UPDATE_IN_PROGRESS	AWS::CloudFormation::Stack	test2	User Initiated
18:12:36 UTC+0550	UPDATE_COMPLETE	AWS::CloudFormation::Stack	test2	
18:12:36 UTC+0550	DELETE_COMPLETE	AWS::ECS::Service	service	
18:10:33 UTC+0550	DELETE_IN_PROGRESS	AWS::ECS::Service	service	
18:10:32 UTC+0550	UPDATE_COMPLETE_CLEANUP_IN_PROGRESS	AWS::CloudFormation::Stack	test2	

Here is the cloudformation template for the JFROG in AWS ECS .

CF template with New VPC&subnets

```
AWSTemplateFormatVersion: 2010-09-09
Description: this is the cloudformation template for Jfrog HA using AWS ECS.
Parameters:

  BucketName:
    Description: Name of MyS3Bucket
    Type: String
    Default: jfro-ecs-storage
  VPC:
    Type: String
  Subnet1:
    Type: String
  Subnet2:
    Type: String

Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: !Ref BucketName
      AccessControl: Private
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
              SSEAlgorithm: AES256


  VPC:
    Type: 'AWS::EC2::VPC'
    Properties:
      CidrBlock: 10.0.0.0/16
```

```
        EnableDnsHostnames: true
        EnableDnsSupport: true
        InstanceTenancy: default
    Subnet1:
      Type: 'AWS::EC2::Subnet'
      Properties:
        AvailabilityZone: !Select
          - 0
          - 'Fn::GetAZs': !Ref 'AWS::Region'
        CidrBlock: !Sub 10.0.0.0/20
        MapPublicIpOnLaunch: true
        VpcId: !Ref VPC
    Subnet2:
      Type: 'AWS::EC2::Subnet'
      Properties:
        AvailabilityZone: !Select
          - 1
          - 'Fn::GetAZs': !Ref 'AWS::Region'
        CidrBlock: !Sub 10.0.32.0/20
        MapPublicIpOnLaunch: true
        VpcId: !Ref VPC
    InternetGateway:
      Type: 'AWS::EC2::InternetGateway'
    VpcGatewayAttachment:
      Type: 'AWS::EC2::VPCGatewayAttachment'
      Properties:
        InternetGatewayId: !Ref InternetGateway
        VpcId: !Ref VPC
    RouteTable:
      Type: 'AWS::EC2::RouteTable'
      Properties:
        VpcId: !Ref VPC
    RouteTableAssociation1:
      Type: 'AWS::EC2::SubnetRouteTableAssociation'
      Properties:
        SubnetId: !Ref Subnet1
        RouteTableId: !Ref RouteTable
    RouteTableAssociation2:
      Type: 'AWS::EC2::SubnetRouteTableAssociation'
      Properties:
        SubnetId: !Ref Subnet2
        RouteTableId: !Ref RouteTable
    InternetRoute:
      Type: 'AWS::EC2::Route'
      DependsOn: VpcGatewayAttachment
      Properties:
        RouteTableId: !Ref RouteTable
        DestinationCidrBlock: 0.0.0.0/0
        GatewayId: !Ref InternetGateway
```

```
ECSRole:
  Type: AWS::IAM::Role
  Properties:
    Path: /
    RoleName: !Sub
      ${ClusterName}-ECSRole-${AWS::Region}
    AssumeRolePolicyDocument:
      Statement:
        - Action:
          - sts:AssumeRole
          Principal:
            Service:
              - ecs-tasks.amazonaws.com
              - ec2.amazonaws.com
              - ecs.amazonaws.com
          Effect: Allow
      Version: 2012-10-17
    ManagedPolicyArns:
      - 'arn:aws:iam::aws:policy/service-role/AmazonEC2RoleforSSM'
      - arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePol
      - arn:aws:iam::aws:policy/service-role/AmazonEC2ContainerServiceforE
    Policies:
      - PolicyName: ecs-service
        PolicyDocument:
          Statement:
            - Effect: Allow
              Action:
                - ecs:ListClusters
                - ecs:ListServices
                - ecs:DescribeServices
                - ecr:ListImages
                - ecs:RegisterTaskDefinition
                - ecs:CreateService
                - ecs:ListTasks
                - ecs:DescribeTasks
                - ecs:CreateService
                - ecs:DeleteService
                - ecs:UpdateService
                - ecs:DescribeContainerInstances
                - ecs:DescribeTaskDefinition
                - application-autoscaling:DescribeScalableTargets
                - iam:ListRoles
              Resource: "*"


 ECSCluster:
  Type: 'AWS::ECS::Cluster'
  Properties:
    ClusterName: prasanna
```

```
taskdefinition:
  Type: 'AWS::ECS::TaskDefinition'
  Properties:
    ExecutionRoleArn: 'arn:aws:iam::536285340728:role/ecsTaskExecutionRole
    ContainerDefinitions:
      - LogConfiguration:
          LogDriver: awslogs
          Options:
            awslogs-group: /ecs/MyJfrog_Task
            awslogs-region: us-east-1
            awslogs-stream-prefix: ecs
        PortMappings:
          - HostPort: 8081
            Protocol: tcp
            ContainerPort: 8081
        Ulimits:
          - Name: nofile
            SoftLimit: 32000
            HardLimit: 32000
        Image: 'docker.bintray.io/jfrog/artifactory-oss:latest'
        Name: JFRO_CONT
    Memory: '4096'
    Family: MyJfrog_Task
    RequiresCompatibilities:
      - FARGATE
    NetworkMode: awsvpc
    Cpu: '2048'
service:
  Type: 'AWS::ECS::Service'
  Properties:
    Cluster: !Ref ECSCluster
    NetworkConfiguration:
      AwsvpcConfiguration:
        AssignPublicIp: ENABLED
        SecurityGroups:
          - sg-0eb21836ffe9d043d
        Subnets:
          - subnet-36f3c919
    DesiredCount: '2'
    LaunchType: FARGATE
    TaskDefinition: !Ref taskdefinition
    LoadBalancers:
      - TargetGroupArn: arn:aws:elasticloadbalancing:us-east-1:53628534072
        ContainerPort: 8081
        ContainerName: JFRO_CONT

LoadBalancer:
  Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
  Properties:
    Name: ecs-services
```

```
      Subnets:
        - !Ref Subnet1
        - !Ref Subnet2
      SecurityGroups:
        - !Ref LoadBalancerSecurityGroup
  LoadBalancerListener:
    Type: 'AWS::ElasticLoadBalancingV2::Listener'
    Properties:
      LoadBalancerArn: !Ref LoadBalancer
      Protocol: HTTP
      Port: 8081
      DefaultActions:
        - Type: forward
          TargetGroupArn: !Ref DefaultTargetGroup
  LoadBalancerSecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupDescription: Security group for loadbalancer to services on ECS
      VpcId: !Ref VPC
      SecurityGroupIngress:
        - CidrIp: 0.0.0.0/0
          IpProtocol: -1
  DefaultTargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      Name: default
      VpcId: !Ref VPC
      Protocol: HTTP
      Port: '8081'

  TargetGroup:
    Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
    Properties:
      Name: books-tg
      VpcId: !Ref VPC
      Port: 8081
      Protocol: HTTP
      Matcher:
        HttpCode: 200-299
      HealthCheckIntervalSeconds: 10
      HealthCheckProtocol: HTTP
      HealthCheckTimeoutSeconds: 5
      HealthyThresholdCount: 10
      TargetType: ip
  ListenerRule:
    Type: 'AWS::ElasticLoadBalancingV2::ListenerRule'
    Properties:
      ListenerArn: !Ref LoadBalancerListener
      Priority: 2
      Actions:
```

```
  - TargetGroupArn: !Ref TargetGroup
    Type: forward
Conditions:
```

```
          - Field: path-pattern
            Values:
              - /
```

}

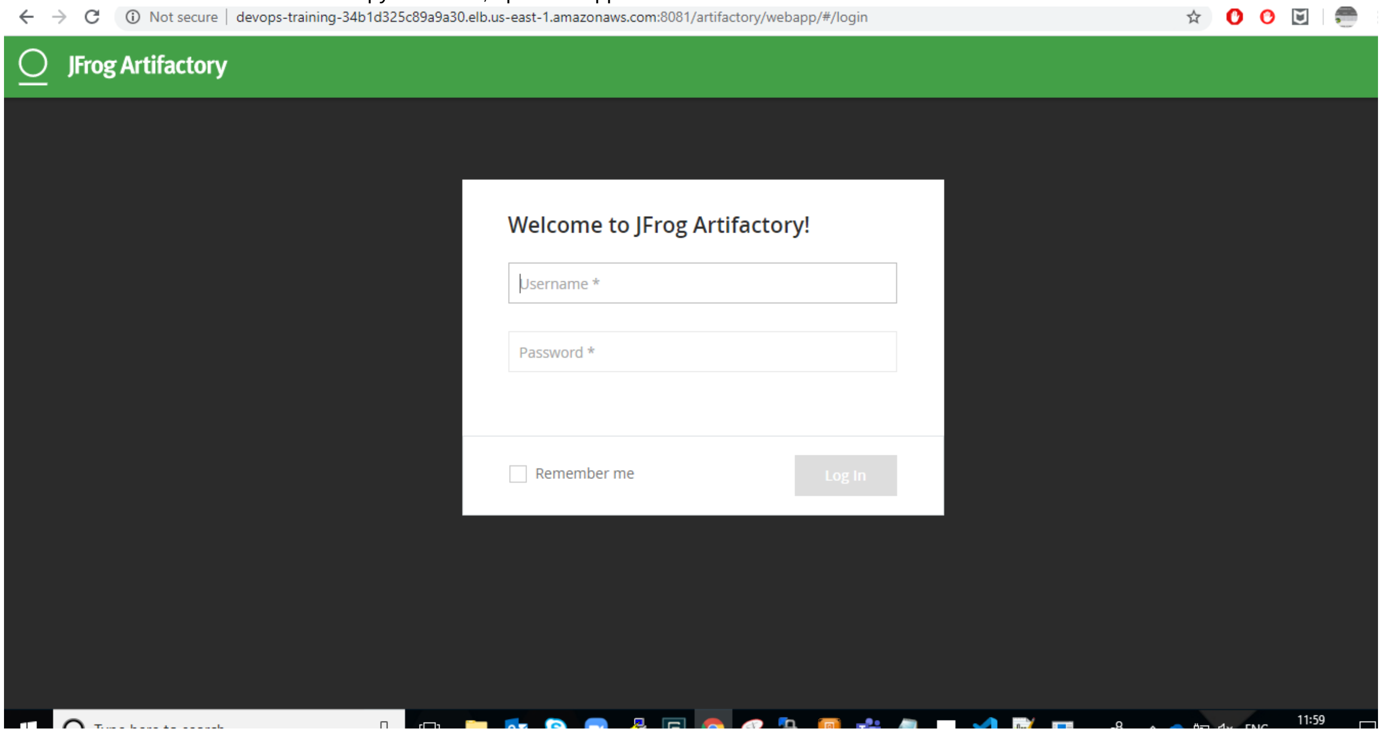
CF template with exist VPC&subnets:

```
AWSTemplateFormatVersion: 2010-09-09
Description: this is the cloudformation template for Jfrog HA using AWS ECS.
Parameters:

  BucketName:
    Description: Name of MyS3Bucket
    Type: String
    Default: jfro-ecs-storage
  VPC:
    Type: String
  Subnet1:
    Type: String
  Subnet2:
    Type: String

Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: !Ref BucketName
      AccessControl: Private
      BucketEncryption:
        ServerSideEncryptionConfiguration:
          - ServerSideEncryptionByDefault:
              SSEAlgorithm: AES256

  ECSRole:
    Type: AWS::IAM::Role
    Properties:
      Path: /
      RoleName: !Sub
        ${ClusterName}-ECSRole-${AWS::Region}
      AssumeRolePolicyDocument:
        Statement:
          - Action:
            - sts:AssumeRole
            Principal:
              Service:
                - ecs-tasks.amazonaws.com
```

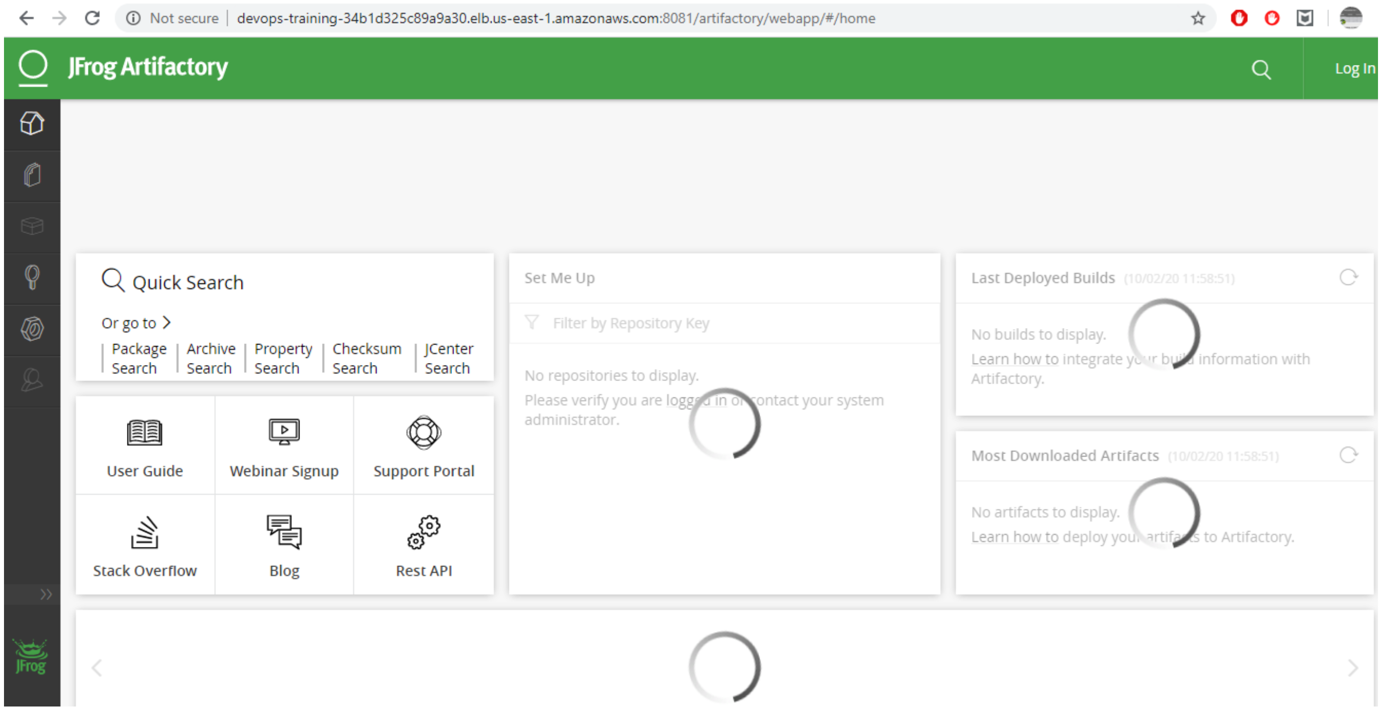
```
                  - ec2.amazonaws.com
                  - ecs.amazonaws.com
              Effect: Allow
          Version: 2012-10-17
        ManagedPolicyArns:
          - 'arn:aws:iam::aws:policy/service-role/AmazonEC2RoleforSSM'
          - arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePol
          - arn:aws:iam::aws:policy/service-role/AmazonEC2ContainerServiceforE
        Policies:
          - PolicyName: ecs-service
            PolicyDocument:
              Statement:
                - Effect: Allow
                  Action:
                    - ecs:ListClusters
                    - ecs:ListServices
                    - ecs:DescribeServices
                    - ecr:ListImages
                    - ecs:RegisterTaskDefinition
                    - ecs:CreateService
                    - ecs:ListTasks
                    - ecs:DescribeTasks
                    - ecs:CreateService
                    - ecs:DeleteService
                    - ecs:UpdateService
                    - ecs:DescribeContainerInstances
                    - ecs:DescribeTaskDefinition
                    - application-autoscaling:DescribeScalableTargets
                    - iam:ListRoles
                  Resource: "*"


     ECSCluster:
      Type: 'AWS::ECS::Cluster'
      Properties:
        ClusterName: prasanna
    taskdefinition:
      Type: 'AWS::ECS::TaskDefinition'
      Properties:
        ExecutionRoleArn: 'arn:aws:iam::536285340728:role/ecsTaskExecutionRole
        ContainerDefinitions:
          - LogConfiguration:
              LogDriver: awslogs
              Options:
                awslogs-group: /ecs/MyJfrog_Task
                awslogs-region: us-east-1
                awslogs-stream-prefix: ecs
            PortMappings:
              - HostPort: 8081
                Protocol: tcp
                ContainerPort: 8081
            Ulimits:
              - Name: nofile
                SoftLimit: 32000
                HardLimit: 32000
            Image: 'docker.bintray.io/jfrog/artifactory-oss:latest'
            Name: JFRO_CONT
        Memory: '4096'
        Family: MyJfrog_Task
        RequiresCompatibilities:
          - FARGATE
        NetworkMode: awsvpc
        Cpu: '2048'
    service:
      Type: 'AWS::ECS::Service'
      Properties:
        Cluster: !Ref ECSCluster
        NetworkConfiguration:
          AwsvpcConfiguration:
            AssignPublicIp: ENABLED
            SecurityGroups:
              - sg-0eb21836ffe9d043d
            Subnets:
              - subnet-36f3c919
        DesiredCount: '2'
        LaunchType: FARGATE
        TaskDefinition: !Ref taskdefinition
        LoadBalancers:
          - TargetGroupArn: arn:aws:elasticloadbalancing:us-east-1:53628534072
            ContainerPort: 8081
            ContainerName: JFRO_CONT

    LoadBalancer:
      Type: 'AWS::ElasticLoadBalancingV2::LoadBalancer'
      Properties:
        Name: ecs-services
        Subnets:
          - !Ref Subnet1
          - !Ref Subnet2
        SecurityGroups:
          - !Ref LoadBalancerSecurityGroup
    LoadBalancerListener:
      Type: 'AWS::ElasticLoadBalancingV2::Listener'
      Properties:
        LoadBalancerArn: !Ref LoadBalancer
        Protocol: HTTP
        Port: 8081
        DefaultActions:
          - Type: forward
            TargetGroupArn: !Ref DefaultTargetGroup
```

```
LoadBalancerSecurityGroup:
  Type: 'AWS::EC2::SecurityGroup'
  Properties:
    GroupDescription: Security group for loadbalancer to services on ECS
    VpcId: !Ref VPC
    SecurityGroupIngress:
      - CidrIp: 0.0.0.0/0
        IpProtocol: -1
DefaultTargetGroup:
  Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
  Properties:
    Name: default
    VpcId: !Ref VPC
    Protocol: HTTP
    Port: '8081'

TargetGroup:
  Type: 'AWS::ElasticLoadBalancingV2::TargetGroup'
  Properties:
    Name: books-tg
    VpcId: !Ref VPC
    Port: 8081
    Protocol: HTTP
    Matcher:
      HttpCode: 200-299
    HealthCheckIntervalSeconds: 10
    HealthCheckProtocol: HTTP
    HealthCheckTimeoutSeconds: 5
    HealthyThresholdCount: 10
    TargetType: ip
ListenerRule:
  Type: 'AWS::ElasticLoadBalancingV2::ListenerRule'
  Properties:
    ListenerArn: !Ref LoadBalancerListener
    Priority: 2
    Actions:
      - TargetGroupArn: !Ref TargetGroup
        Type: forward
    Conditions:
      - Field: path-pattern
        Values:
          - /
```

1. Go to the loadbalancer and copy the DNS , open the application as below.



devops-training-34b1d325c89a9a30.elb.us-east-1.amazonaws.com:8081/artifactory/webapp/#/login
JFrog Artifactory
Welcome to JFrog Artifactory!
Username *
Password *
Remember me
Log In



devops-training-34b1d325c89a9a30.elb.us-east-1.amazonaws.com:8081/artifactory/webapp/#/home
JFrog Artifactory
Log In
Quick Search
Or go to
Package Search
Archive Search
Property Search
Checksum Search
JCenter Search
Set Me Up
Filter by Repository Key
No repositories to display.
Please verify you are logged in or contact your system administrator.
Last Deployed Builds
No builds to display.
Learn how to integrate your build information with Artifactory.
Most Downloaded Artifacts
No artifacts to display.
Learn how to deploy your artifacts to Artifactory.
User Guide
Webinar Signup
Support Portal
Stack Overflow
Blog
Rest API

thats all for jfrog HA, Now try to delete one container in cluster, see the changes automatically a new container up and running. Now with this we have achieved Jfrog HA.

Troubleshooting:

1. Please update the correct docker image and upload in ECR only, then try to deploy in cloud-formation template.

2. Integrate the NLB loadbalancer with Jfrog cluster by integrating the correct service in the cluster.
3. For any issues related to cluster, try to check the events of the service and logs, try to debugg based on issues we faced.
4. Use same type of cluster and task definition i.e ec2/fargate.
5. Develop the correct AWS modules based on requirement.
6. Map the correct resources using ref in the modules resource section.
7. Try to put the conditions for dependency resources.
8. Pass the parameters for key pairs deployment.
9. Structure the resources in sequences, it is easy to under the workflow.
10. Check the logs of the service for container logs.
11. If there is any dependencies in docker image, need to pass in the dockerfile and build new one, and update the template.
12. Try to use the update stack option, to avoid newly created stacks.
13. Use CI/CD for continuous deployment of template.

Thank you.