Full infrastructure as code and CI/CD using AWS Elastic Container Service

who needs Kubernetes?

Mark van Holsteijn – CTO

ACTIEF ALS TELECOMAANBIEDER?

COIN helpt je verder

U bent actief als telecomaanbieder. Dat betekent dat u moet voldoen aan een aantal wettelijke verplichtingen. Het aanbieden van nummerbehoud en nummerafscherming aan uw klanten zijn enkele voorbeelden daarvan. Maar wat is wel en niet verplicht en hoe regelt u deze zaken? Vereniging COIN helpt u snel en goed op weg met een aantal praktische diensten.

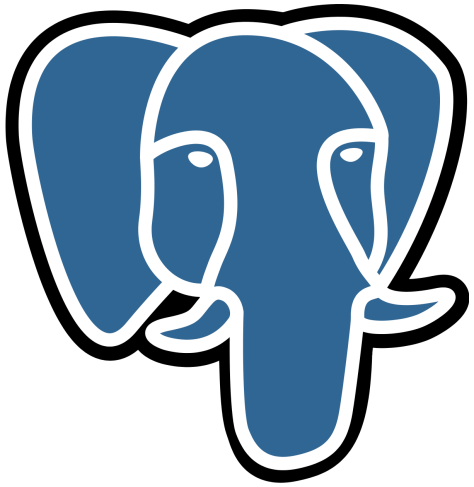
Wie is COIN?

NUMMERPORTABILITEIT

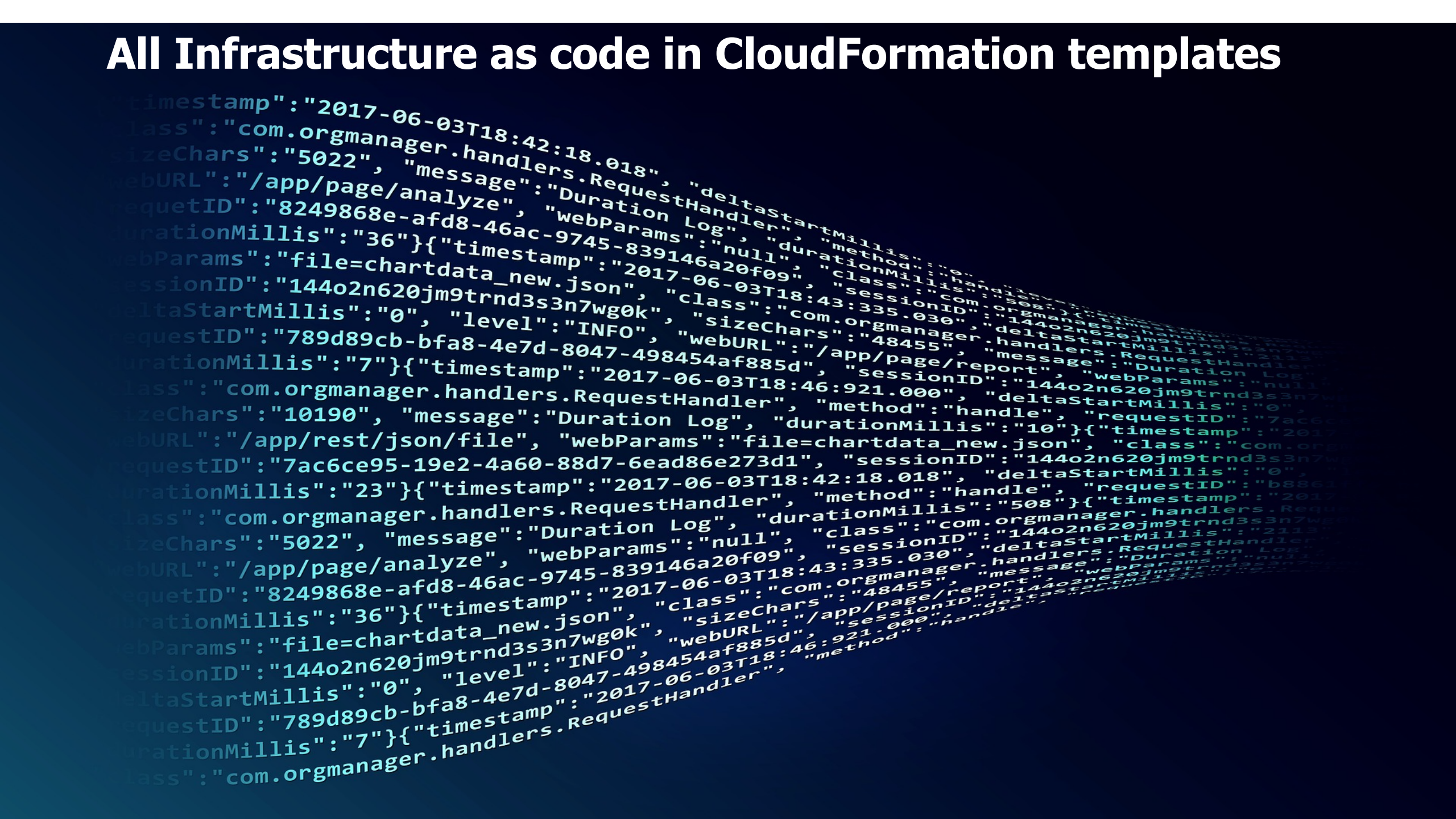
OVERSTAPPEN

Full Automated Creation of Cloud Application Landscape - No Manual Intervention Needed

The Landscape components

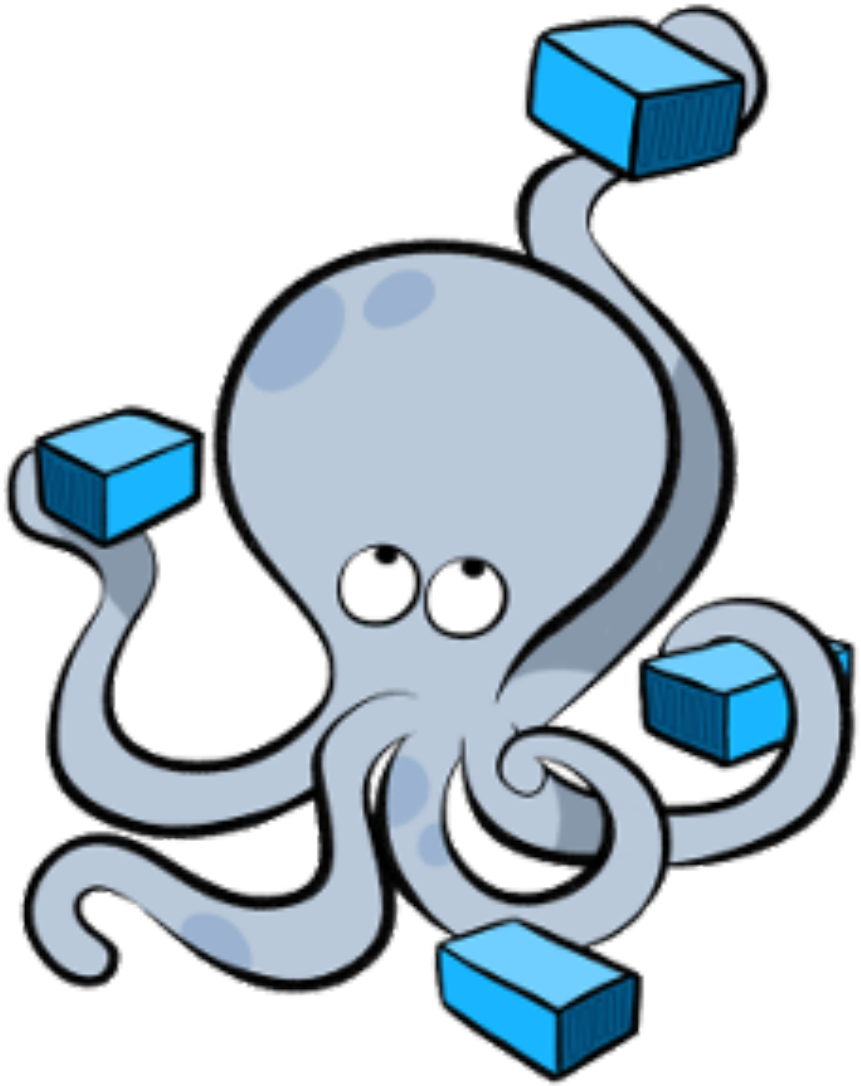


All Infrastructure as code in CloudFormation templates



All applications packaged into Docker Images

Docker Compose creates a complete Integration Environment on the desktop



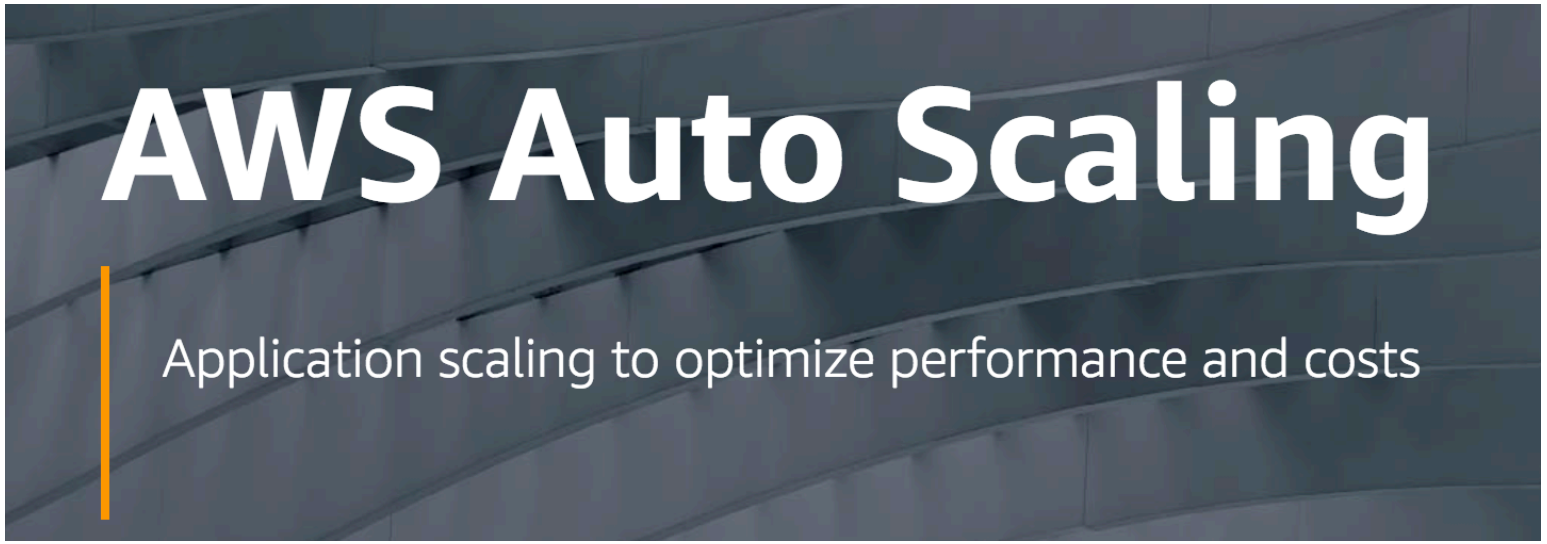

All secrets randomly generated and stored in AWS Parameter Store

All secrets randomly generated and stored in AWS Parameter Store

```
DBPassword:
  Type: Custom::Secret
  Properties:
    Name: !Sub '/${AWS::StackName}/oracle/root/password'
    Alphabet: abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
    MaxLength: 30
    ReturnSecret: true
```

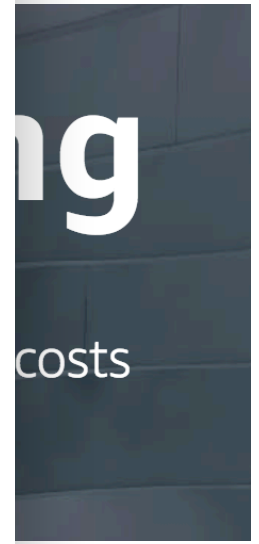
https://github.com/binxio/cfn-secret-provider

CI Runners are stateless in auto scaling groups

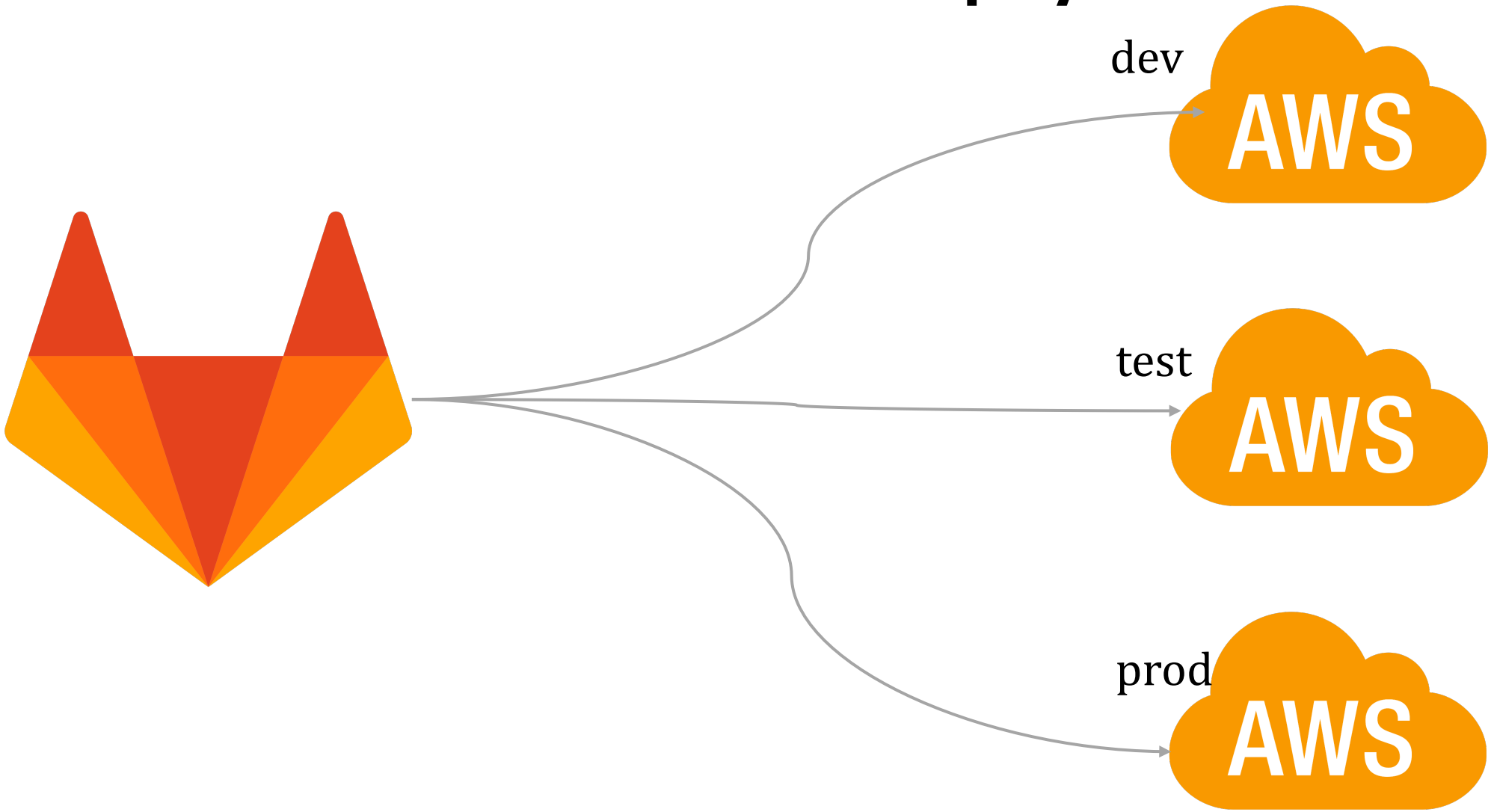


CI Runners are stateless in auto scaling groups

```
LaunchConfiguration:
  Type: AWS::AutoScaling::LaunchConfiguration
  DependsOn: PrivateKey
  Properties:
    ImageId: !Ref 'AMI'
    SecurityGroups:
      - !Ref 'VpcSecurityGroup'
    InstanceType: 't3.small'
    IamInstanceProfile: !Ref 'InstanceProfile'
    UserData:
      Fn::Base64: |
        #!/bin/bash
        export AWS_DEFAULT_REGION=eu-west-1
        export DEFAULT_IMAGE=gitlab-runner:0.1.2


        ## register the gitlab runner
        mkdir /etc/gitlab-runner
        docker run -v "/etc/gitlab-runner:/etc/gitlab-runner" \
              gitlab/gitlab-runner:latest register \
                      -n --url "https://gitlab.com" \
                      --description "aws runner" \
                      --executor "docker" \
                      --docker-image "$DEFAULT_IMAGE" \
                      --docker-volumes /home/gitlab/.ssh:/home/gitlab-runner/.ssh \
                      --registration-token \
                              "$(aws --query Parameter.Value --output text ssm get-parameter --name /git
```



Gitlab runners assume role to deploy to AWS accounts



Gitlab runners assume role to deploy to AWS accounts

```yaml
CiCdRole:
  Type: AWS::IAM::ManagedPolicy
  Properties:
    ManagedPolicyName: CiCdRole
    Description: allows CI/CD runners to deploy to test, acceptance, production and devops
    PolicyDocument:
      Version: '2012-10-17'
      Statement:
          - Effect: Allow
            Action:
              - sts:AssumeRole
            Resource:
              - !Sub 'arn:aws:iam::${DevOpsAccountId}:role/CloudFormationStackUpdater'
              - !Sub 'arn:aws:iam::${TestAccountId}:role/CloudFormationStackUpdater'
              - !Sub 'arn:aws:iam::${AcceptanceAccountId}:role/CloudFormationStackUpdater'
              - !Sub 'arn:aws:iam::${ProductionAccountId}:role/CloudFormationStackUpdater'
          - Effect: Allow
            Action:
              - s3:*
```


```yaml
deploy_test:
  stage: deploy
  before_script:
  script:
    - assume_role -p test -a 111111111111 -r CloudFormationStackUpdater -R gitlab-runner
    - sceptre --var-file environments/test.yaml launch-env web
```

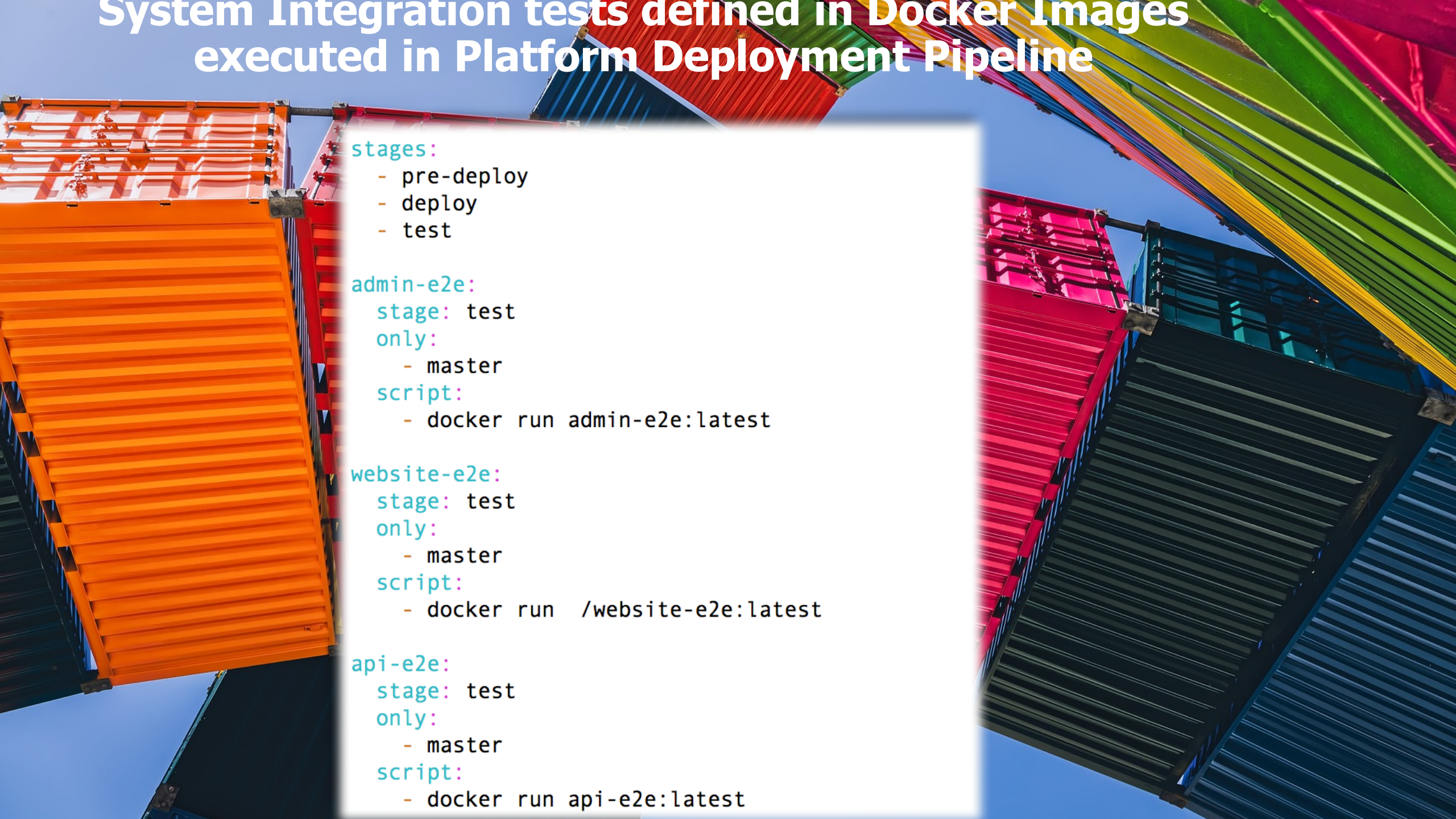
CI/CD of all users, groups, roles and policies to AWS accounts

CI/CD of all users, groups, roles and policies to AWS accounts

```yaml
AcceptanceAdministratorRole:
  Type: AWS::IAM::ManagedPolicy
  Properties:
    ManagedPolicyName: AllowAdministratorRoleOnAcceptance
    Description: allows users to assume administrator role in acceptance accounts.
    Groups:
      - !Ref Administrators
    PolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Action:
            - sts:AssumeRole
          Resource:
            - !Sub 'arn:aws:iam::${AtAccountId}:role/Administrator'
          Condition:
            Bool:
              aws:MultiFactorAuthPresent: 'true'
```

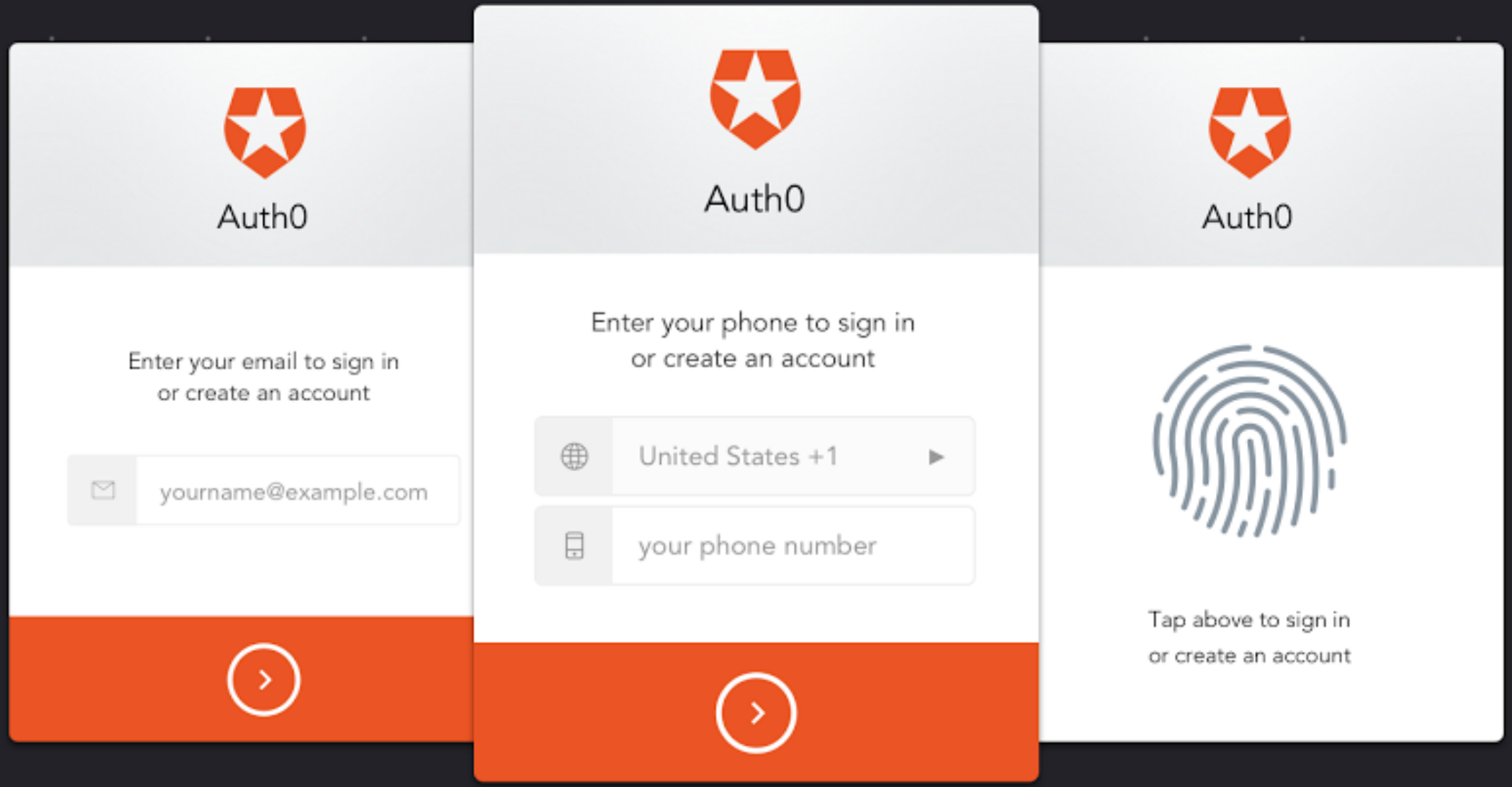
System Integration tests defined in Docker Images executed in Platform Deployment Pipeline

System Integration tests defined in Docker Images executed in Platform Deployment Pipeline



```yaml
stages:
  - pre-deploy
  - deploy
  - test

admin-e2e:
  stage: test
  only:
    - master
  script:
    - docker run admin-e2e:latest

website-e2e:
  stage: test
  only:
    - master
  script:
    - docker run  /website-e2e:latest

api-e2e:
  stage: test
  only:
    - master
  script:
    - docker run api-e2e:latest
```

CI/CD of Auth0 Configuration with the Application

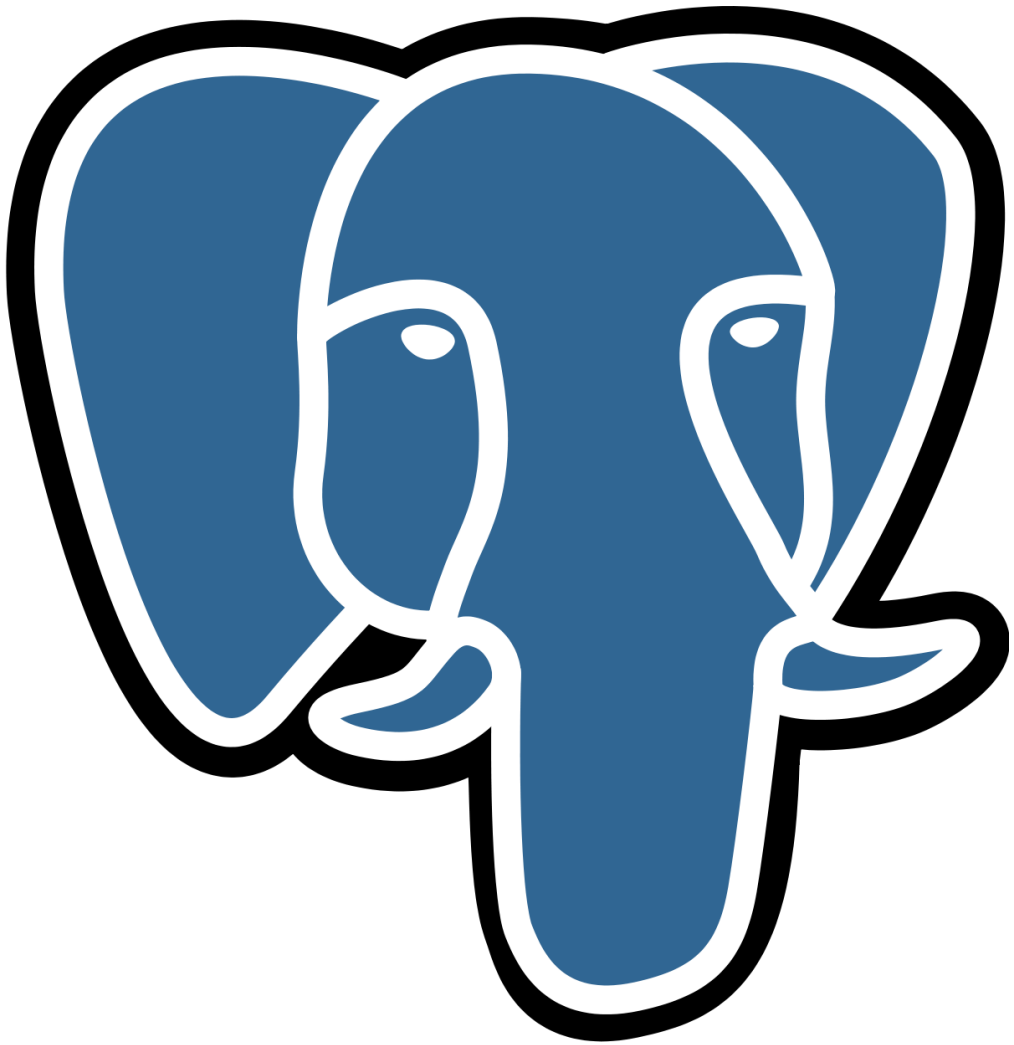


CI/CD of Auth0 Configuration with the Application

```yaml
Auth0API:
  Type: Custom::Auth0ResourceServer
  Properties:
    ServiceToken: !Sub 'arn:aws:lambda:${AWS::Region}:${AWS::AccountId}:function:cf
    Value:
      name: consumer-api
      identifier: https://api.binx.io/consumer-api
      allow_offline_access: false
      skip_consent_for_verifiable_first_party_clients: true
      token_lifetime: 900
      token_lifetime_for_web: 900
      signing_alg: RS256
      scopes:
        - value: manage:consumers
          description: manage all consumers
        - value: manage:own-consumers
          description: manage own consumers
        - value: manage:users
          description: manage users
        - value: manage:groups
          description: manage groups
        - value: view:access
          description: can view the application
        - value: query:eventlog
          description: can query the event logs
```

https://github.com/binxio/cfn-auth0-provider

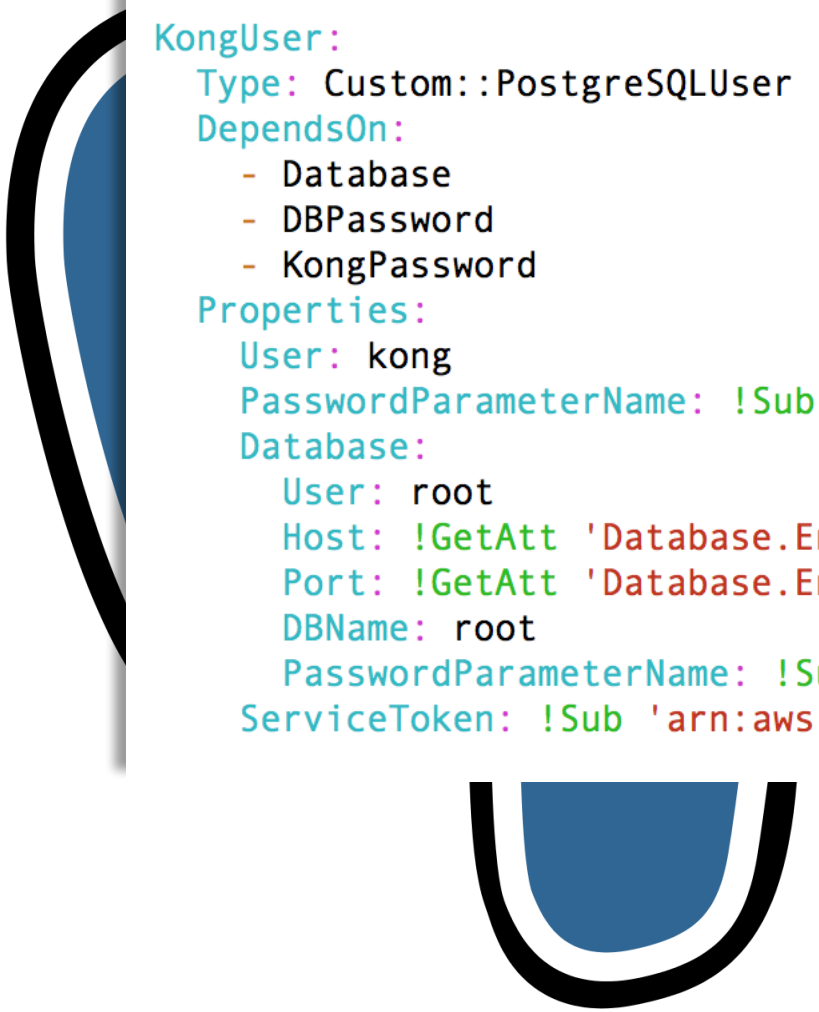
CI/CD of PostgreSQL and Oracle schemas and users



ORACLE®

CI/CD of PostgreSQL and Oracle schemas and users

```
KongUser:
  Type: Custom::PostgreSQLUser
  DependsOn:
    - Database
    - DBPassword
    - KongPassword
  Properties:
    User: kong
    PasswordParameterName: !Sub '/${AWS::StackName}/postgres/kong/PGPASSWORD'
    Database:
      User: root
      Host: !GetAtt 'Database.Endpoint.Address'
      Port: !GetAtt 'Database.Endpoint.Port'
      DBName: root
      PasswordParameterName: !Sub '/${AWS::StackName}/postgres/root/PGPASSWORD'
    ServiceToken: !Sub 'arn:aws:lambda:${AWS::Region}:${AWS::AccountId}:function:binxio
```



ORACLE®

https://github.com/binxio/cfn-postgresql-provider

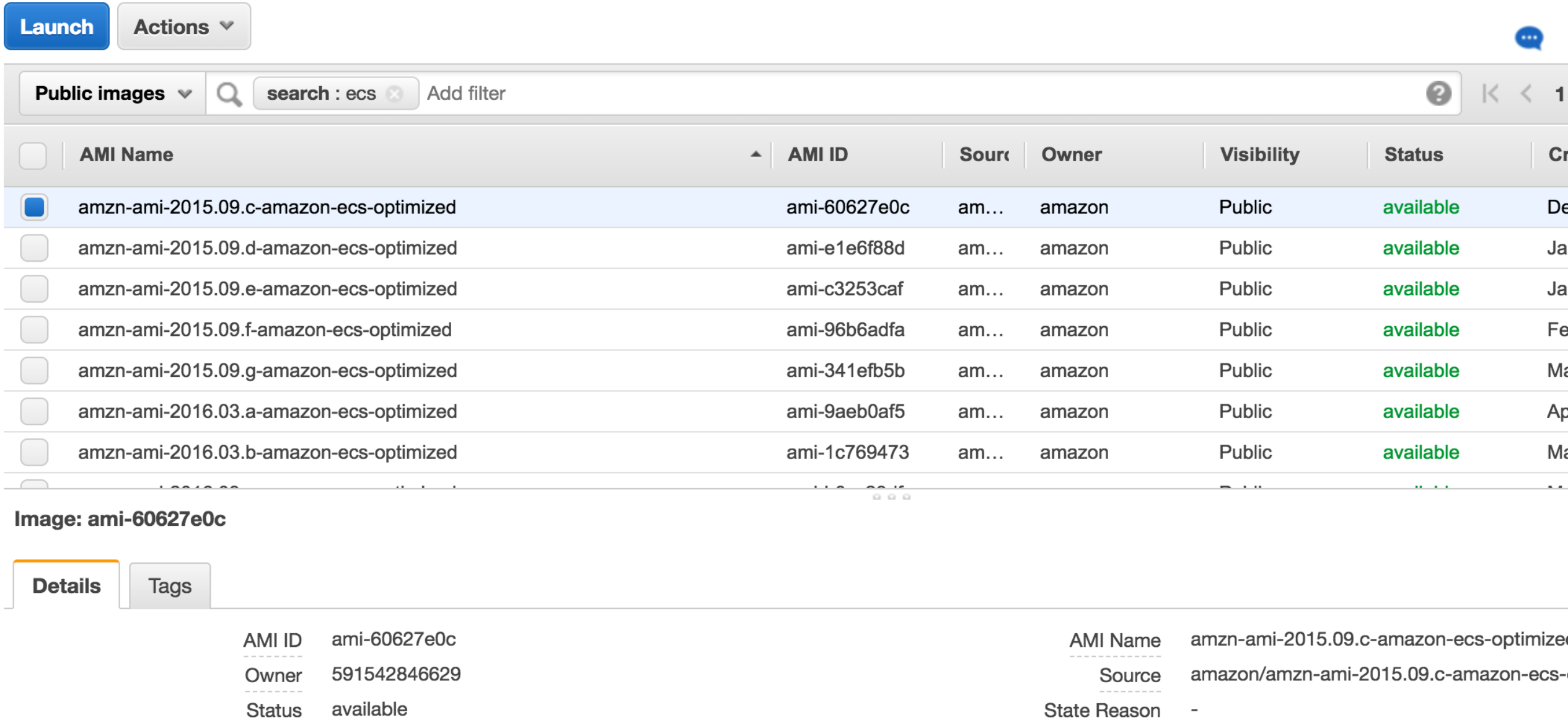
CI/CD of Kong API Gateway Configuration with Application

CI/CD of Kong API Gateway Configuration with Application

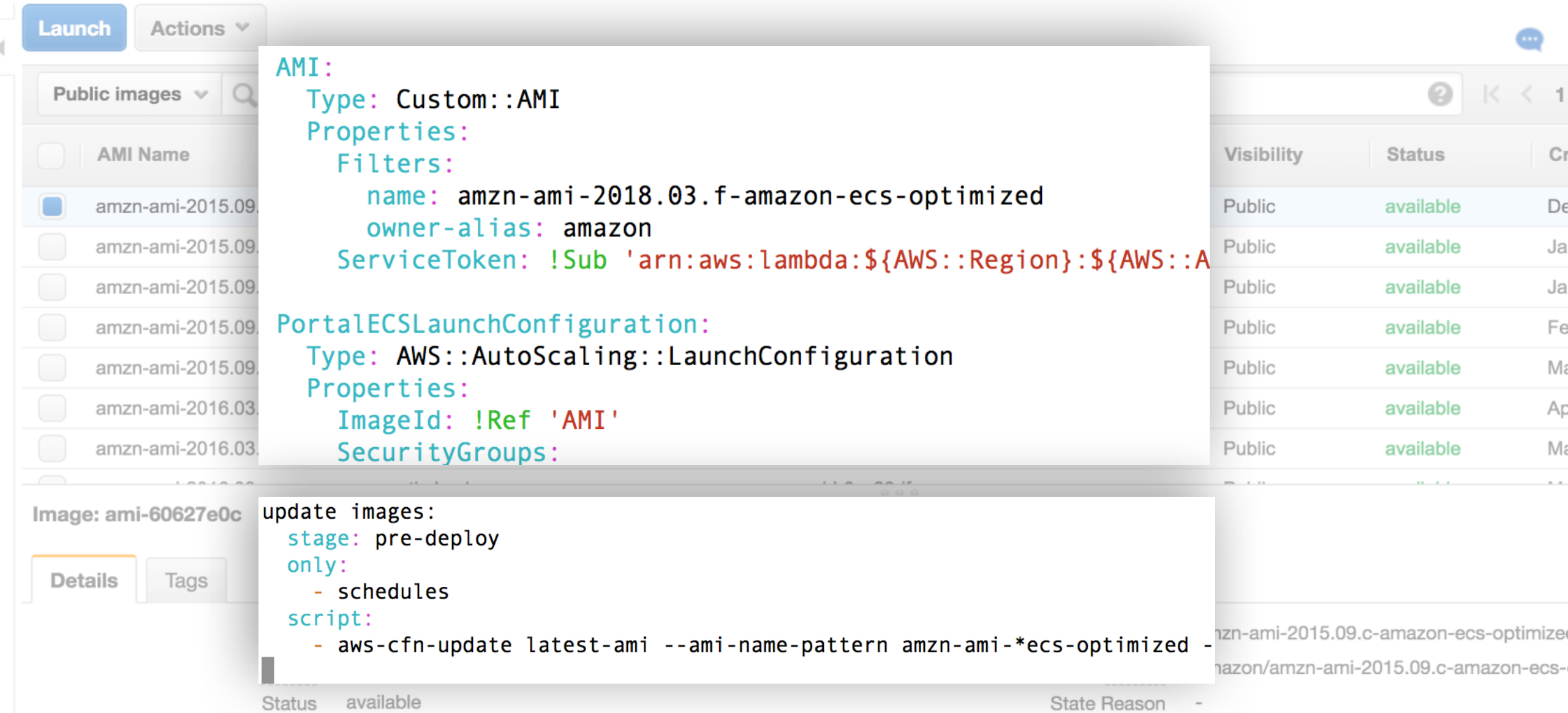
```yaml
Resources:
  HeaderService:
    Type: Custom::KongService
    Properties:
      Service:
        name: header-service
        host: httpbin.org
        protocol: https
      AdminURL: !Ref 'AdminURL'
      ServiceToken: !Sub 'arn:aws:lambda:${AWS::Region}:${AWS::AccountId}:fun
  HeaderRoute:
    Type: Custom::KongRoute
    Properties:
      Route:
        paths:
          - /headers
        service:
          id: !Ref 'HeaderService'
      AdminURL: !Ref 'AdminURL'
      ServiceToken: !Sub 'arn:aws:lambda:${AWS::Region}:${AWS::AccountId}:fun
```

https://github.com/binxio/cfn-kong-provider

Automated updates of used Amazon Machine Images



Automated updates of used Amazon Machine Images

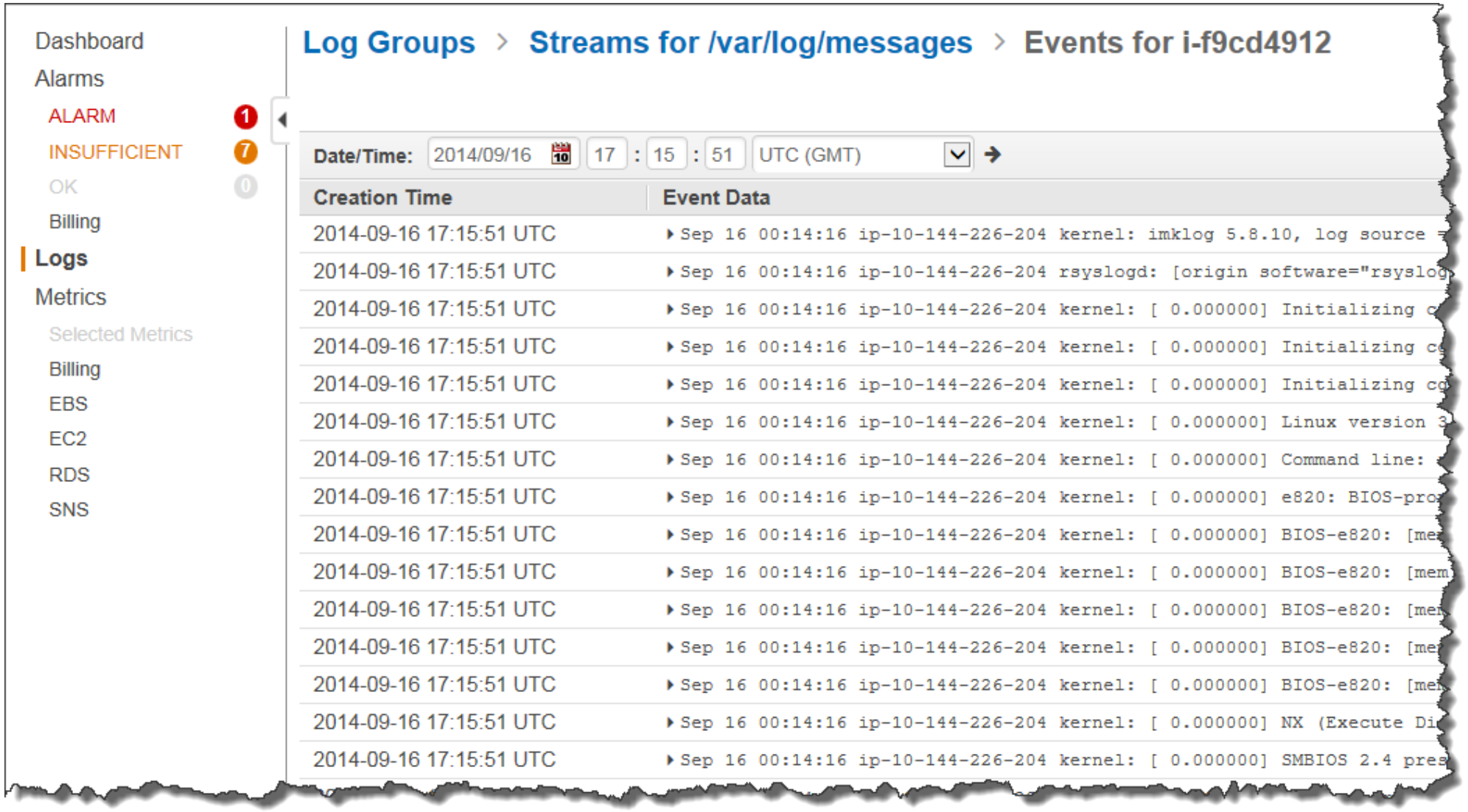


Rolling update of ECS server instances – Without downtime

Rolling update of ECS server instances – Without downtime

```
ASGSNSTopic:
  Type: AWS::SNS::Topic
  Properties:
    Subscription:
      - Endpoint: !GetAtt 'LambdaFunctionForASG.Arn'
        Protocol: lambda
  DependsOn: LambdaFunctionForASG

LambdaFunctionForASG:
  Type: AWS::Lambda::Function
  Properties:
    Code:
      S3Bucket: system-lambdas-eu-central-1
      S3Key: lambdas/cfn-ecs-lifecycle-service-0.1.3.zip
    Description: Lambda code for the autoscaling hook triggers invoked when autoscaling
      events of launching and terminating instance occur
    Handler: cfn_ecs_lifecycle_service.handler
    Role: !GetAtt 'LambdaExecutionRole.Arn'
    Runtime: python2.7
    Timeout: '300'
```

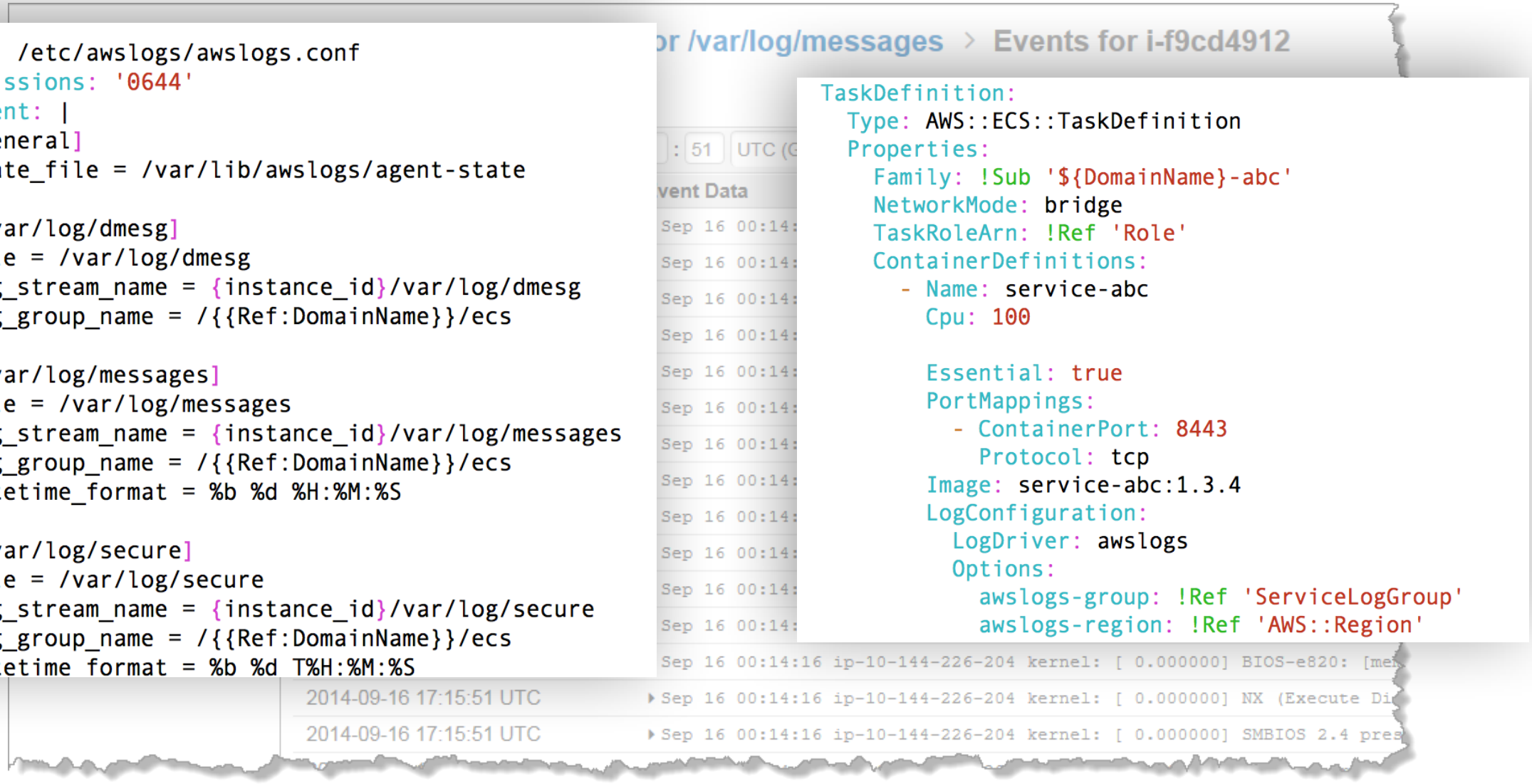
All System and Application Logging to CloudWatch Logs



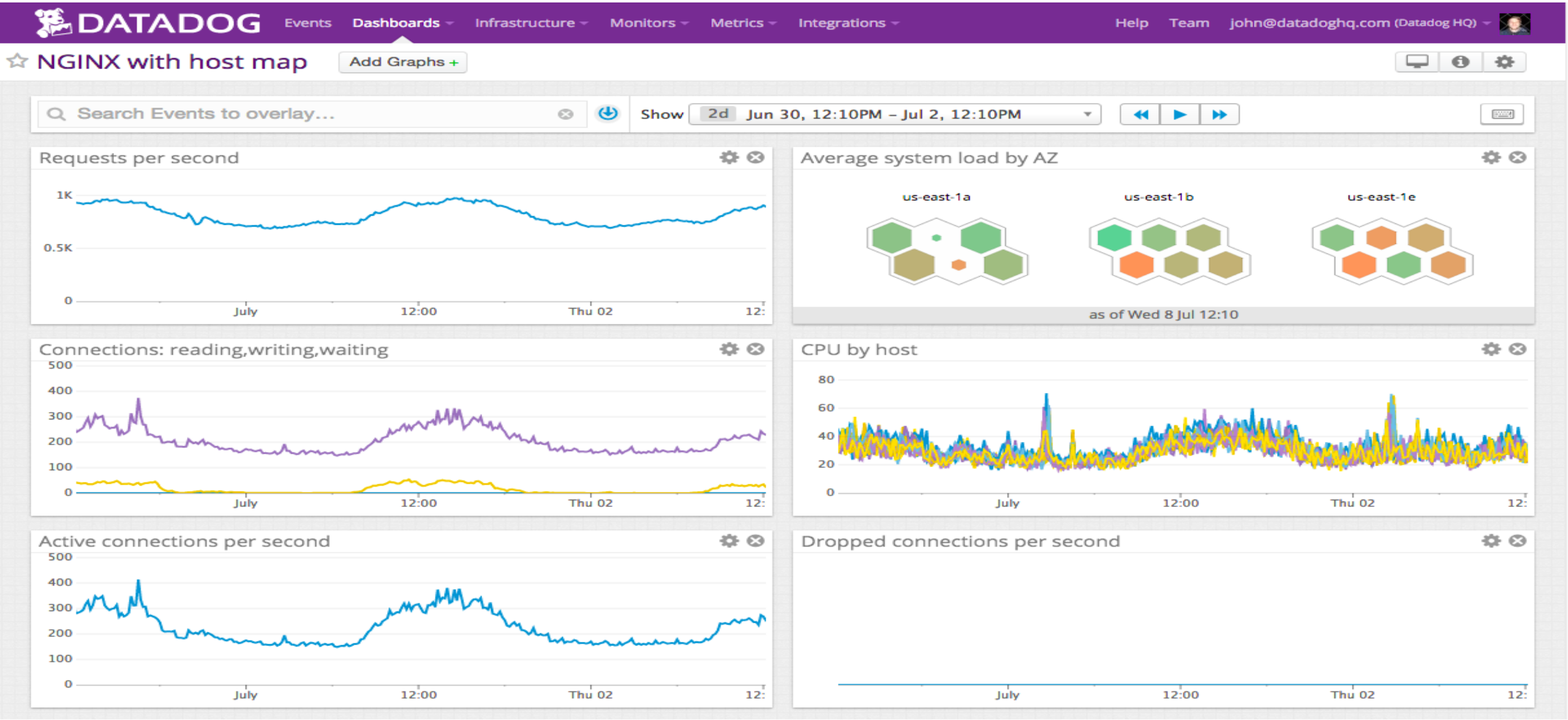
All System and Application Logging to CloudWatch Logs

```
- path: /etc/awslogs/awslogs.conf
  permissions: '0644'
  content: |
    [general]
    state_file = /var/lib/awslogs/agent-state

    [/var/log/dmesg]
    file = /var/log/dmesg
    log_stream_name = {instance_id}/var/log/dmesg
    log_group_name = /{{Ref:DomainName}}/ecs

    [/var/log/messages]
    file = /var/log/messages
    log_stream_name = {instance_id}/var/log/messages
    log_group_name = /{{Ref:DomainName}}/ecs
    datetime_format = %b %d %H:%M:%S

    [/var/log/secure]
    file = /var/log/secure
    log_stream_name = {instance_id}/var/log/secure
    log_group_name = /{{Ref:DomainName}}/ecs
    datetime_format = %b %d T%H:%M:%S
```

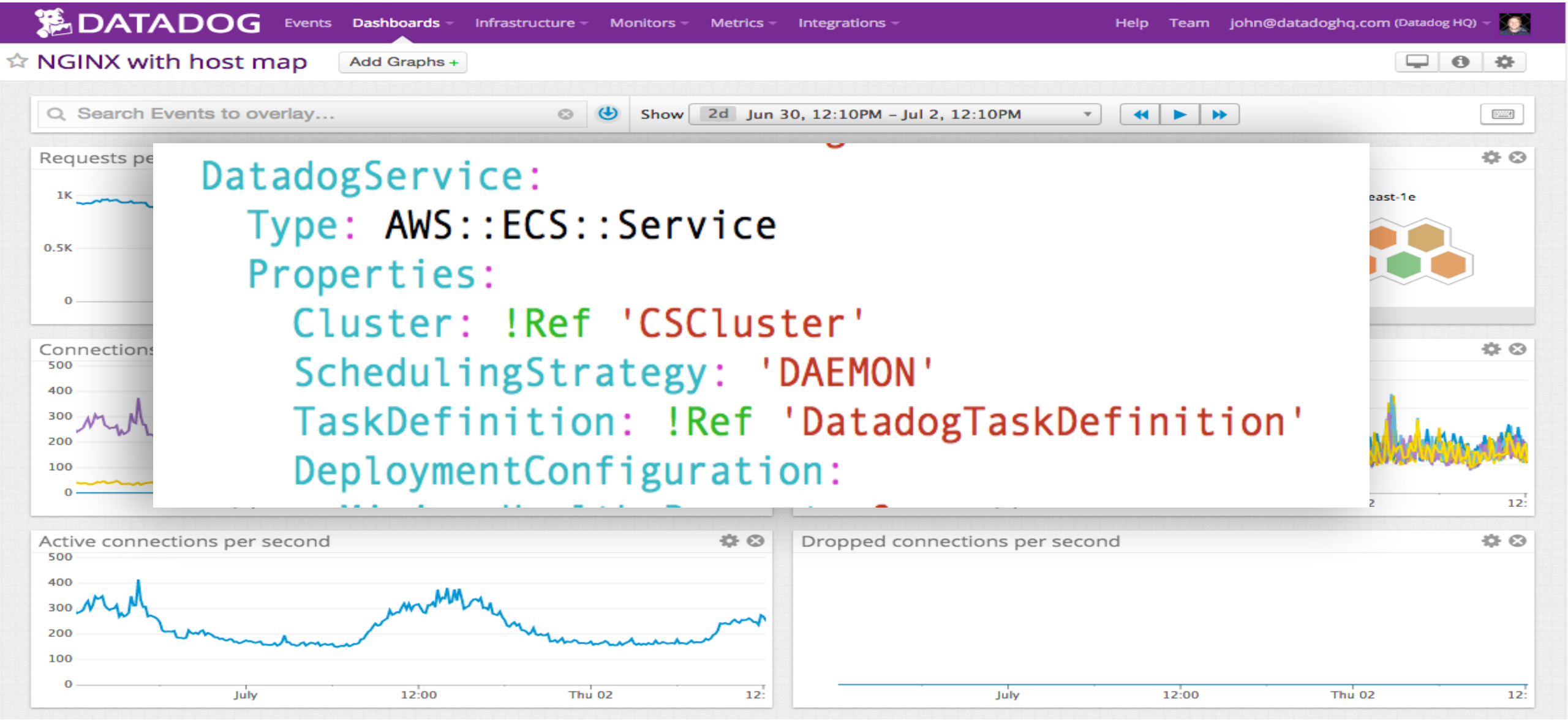
```
TaskDefinition:
  Type: AWS::ECS::TaskDefinition
  Properties:
    Family: !Sub '${DomainName}-abc'
    NetworkMode: bridge
    TaskRoleArn: !Ref 'Role'
    ContainerDefinitions:
      - Name: service-abc
        Cpu: 100

        Essential: true
        PortMappings:
          - ContainerPort: 8443
            Protocol: tcp
        Image: service-abc:1.3.4
        LogConfiguration:
          LogDriver: awslogs
          Options:
            awslogs-group: !Ref 'ServiceLogGroup'
            awslogs-region: !Ref 'AWS::Region'
```



Datadog used as metrics collector and alert monitor



Datadog used as metrics collector and alert monitor



Complete Infrastructure As Code setup replicated at the Harbor of Rotterdam in less than 7 days

❖ While replacing Gitlab with Github &AWS CodeBuild and Auth0 with a AWS MFA setup

Full infrastructure as code and CI/CD using AWS Elastic Container Service