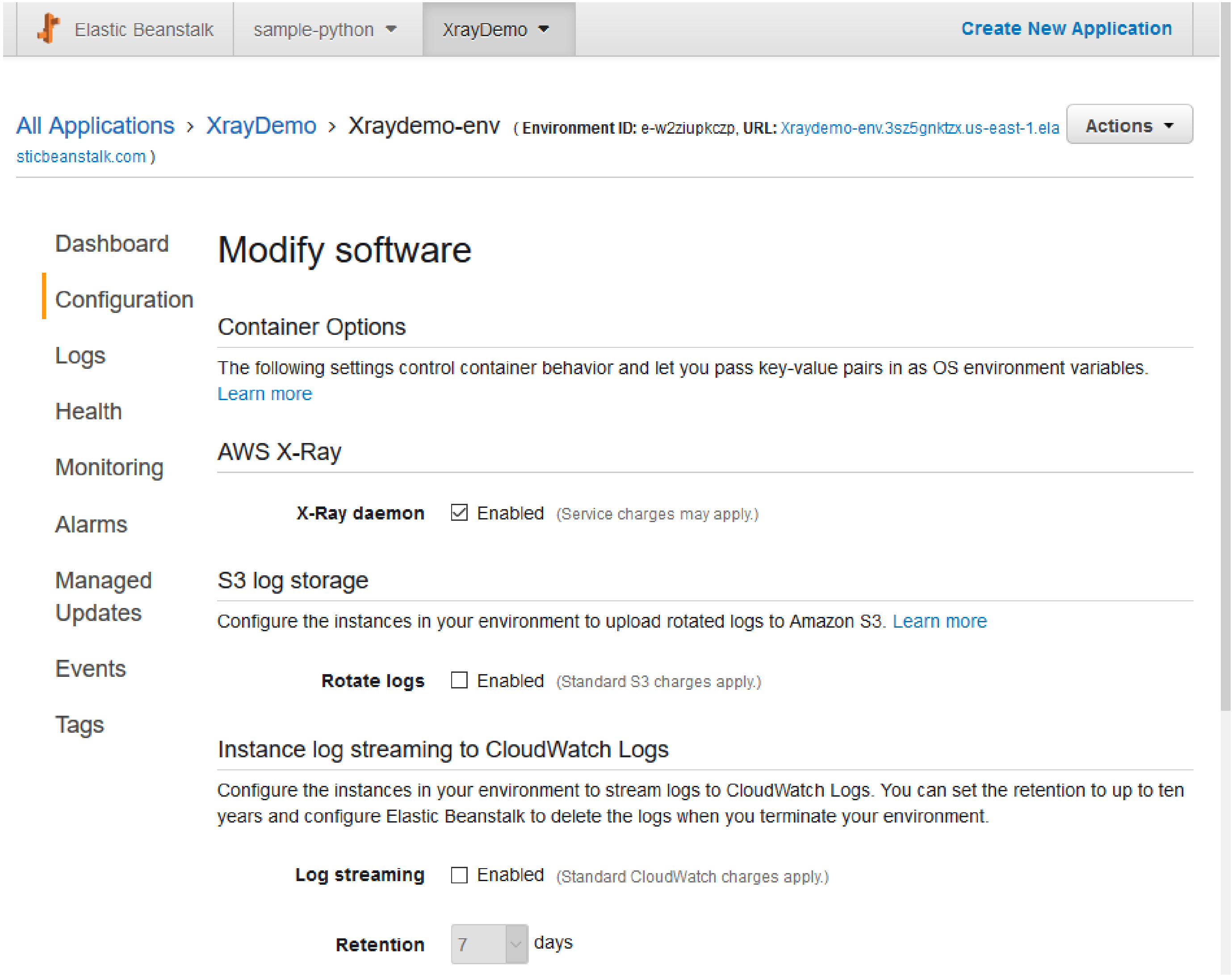
AWS XRay Integration with different AWS services

XRay Integration with different AWS services-

1.Elastic Beanstalk

To enable the X-Ray daemon in the Elastic Beanstalk console

1. Open the Elastic Beanstalk console.
2. Navigate to the management console for your environment.
3. Choose **Configuration**.
4. Choose **Software Settings**.
5. For **X-Ray daemon**, choose **Enabled**.
6. Choose **Apply**.



All Applications › XrayDemo › Xraydemo-env (**Environment ID:** e-w2ziupkczp, **URL:** Xraydemo-env.3sz5gnktzx.us-east-1.elasticbeanstalk.com)

Modify software

Container Options

The following settings control container behavior and let you pass key-value pairs in as OS environment variables. Learn more

AWS X-Ray

X-Ray daemon ☑ Enabled (Service charges may apply.)

S3 log storage

Configure the instances in your environment to upload rotated logs to Amazon S3. Learn more

Rotate logs ☐ Enabled (Standard S3 charges apply.)

Instance log streaming to CloudWatch Logs

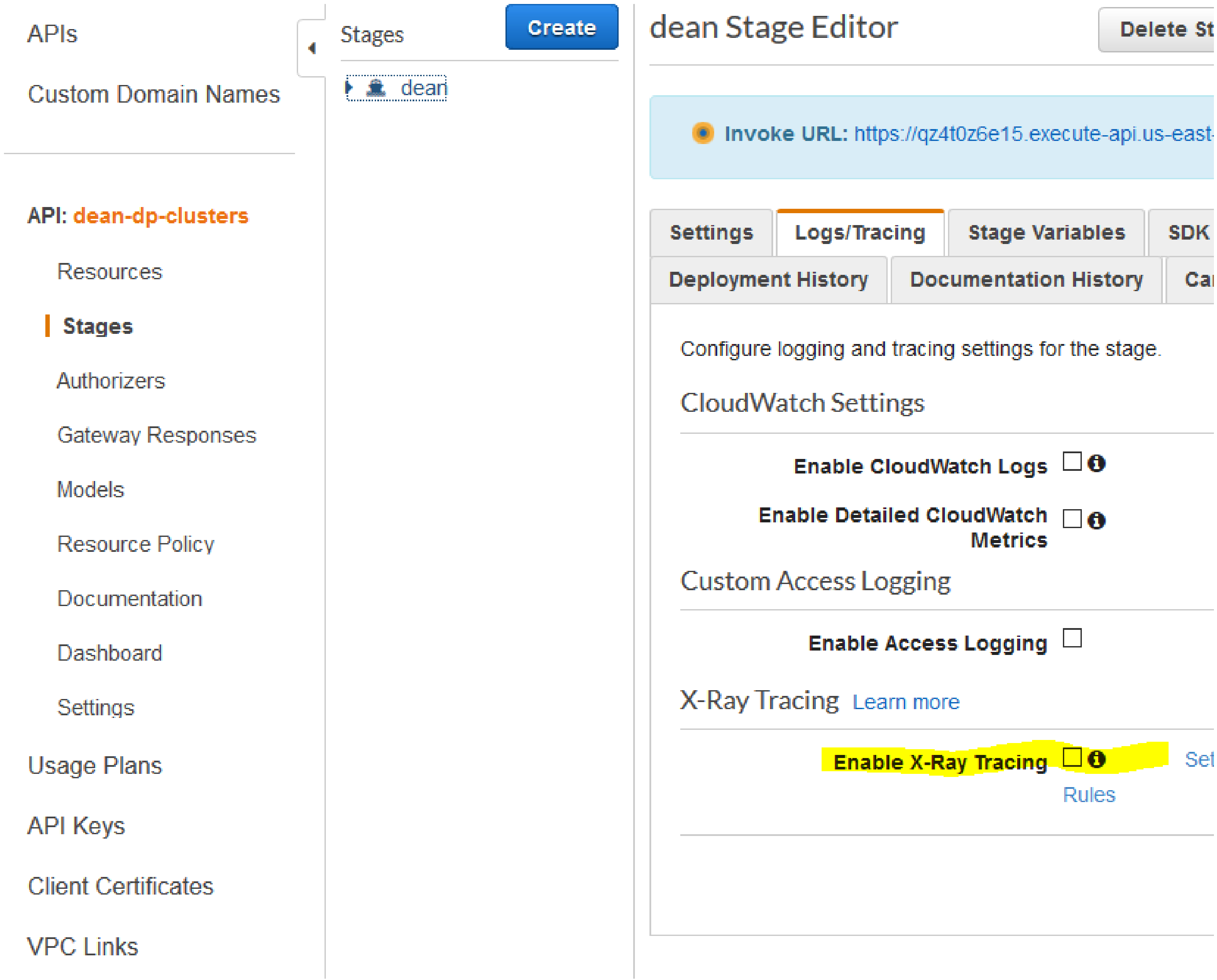
Configure the instances in your environment to stream logs to CloudWatch Logs. You can set the retention to up to ten years and configure Elastic Beanstalk to delete the logs when you terminate your environment.

Log streaming ☐ Enabled (Standard CloudWatch charges apply.)

Retention 7 days

2.API Gateway

Enabling Xray for API Gateway,



dean Stage Editor

Invoke URL: https://qz4t0z6e15.execute-api.us-east-

Configure logging and tracing settings for the stage.

CloudWatch Settings

Enable CloudWatch Logs □

Enable Detailed CloudWatch Metrics □

Custom Access Logging

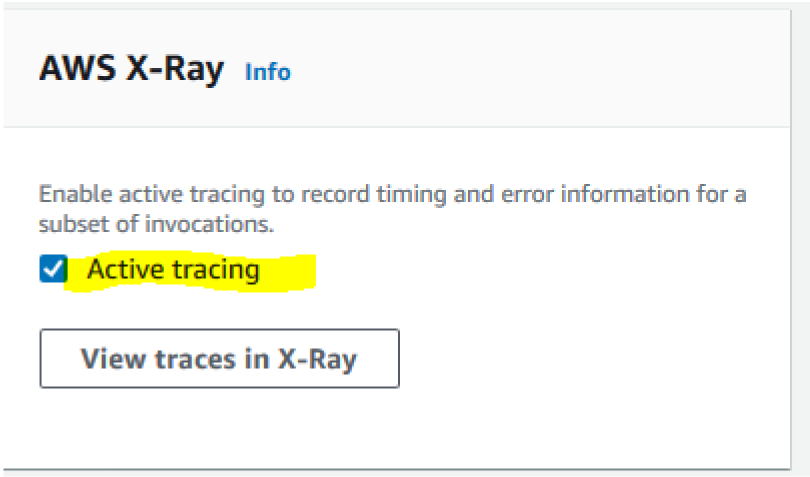
Enable Access Logging □

X-Ray Tracing

Enable X-Ray Tracing □

3.Lambda

Enabling Xray for Lambda-



AWS X-Ray

Enable active tracing to record timing and error information for a subset of invocations.

☑ Active tracing

4.EC2

Enabling xray with EC2 instances,

Use a user data script to run the daemon automatically when you launch the instance.

Linux EC2 Instance-

```
#!/bin/bash
curl https://s3.dualstack.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/xray-daemon/aws-xray-daemon-3.x.rpm -o /home/ec2-user/xray.rpm
yum install -y /home/ec2-user/xray.rpm
```

Windows EC2 instance,

```
<powershell>
if ( Get-Service "AWSXRayDaemon" -ErrorAction SilentlyContinue ) {
sc.exe stop AWSXRayDaemon
sc.exe delete AWSXRayDaemon
}

$targetLocation = "C:\Program Files\Amazon\XRay"
if ((Test-Path $targetLocation) -eq 0) {
mkdir $targetLocation
}

$zipFileName = "aws-xray-daemon-windows-service-3.x.zip"
$zipPath = "$targetLocation\$zipFileName"
$destPath = "$targetLocation\aws-xray-daemon"
if ((Test-Path $destPath) -eq 1) {
Remove-Item -Recurse -Force $destPath
}

$daemonPath = "$destPath\xray.exe"
$daemonLogPath = "$targetLocation\xray-daemon.log"
$url = "https://s3.dualstack.us-west-2.amazonaws.com/aws-xray-assets.us-west-2/xray-daemon/aws-xray-daemon-windows-service-3.x.zip"

Invoke-WebRequest -Uri $url -OutFile $zipPath
Add-Type -Assembly "System.IO.Compression.Filesystem"
[io.compression.zipfile]::ExtractToDirectory($zipPath, $destPath)

New-Service -Name "AWSXRayDaemon" -StartupType Automatic -BinaryPathName "`"$daemonPath`" -f `"$daemonLogPath`""
sc.exe start AWSXRayDaemon
</powershell>
```

5.ECS

Enabling xray with ECS service,

In Amazon ECS, create a Docker image that runs the X-Ray daemon, upload it to a Docker image repository, and then deploy it to your Amazon ECS cluster. You can use port mappings and network mode settings in your task definition file to allow your application to communicate with the daemon container.

Example Dockerfile – Ubuntu

For Debian derivatives, you also need to install certificate authority (CA) certificates to avoid issues when downloading the installer.

```
FROM ubuntu:16.04

RUN apt-get update && apt-get install -y --force-yes --no-install-recommends apt-transport-https curl ca-certificates wget && apt-get clean && apt-get autoremove && rm -rf /var/lib/apt/lists/*

RUN wget https://s3.dualstack.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/xray-daemon/aws-xray-daemon-3.x.deb

RUN dpkg -i aws-xray-daemon-3.x.deb

ENTRYPOINT ["/usr/bin/xray", "--bind=0.0.0.0:2000", "--bind-tcp=0.0.0.0:2000"]

EXPOSE 2000/udp

EXPOSE 2000/tcp
```

In your task definition, the configuration depends on the networking mode that you use. Bridge networking is the default and can be used in your default VPC. In a bridge network, publish UDP port 2000, and create a link from your application container to the daemon container. Use the AWS_XRAY_DAEMON_ADDRESS environment variable to tell the X-Ray SDK where to send traces.

Example Task definition

```
{
"name": "xray-daemon",
"image": "123456789012.dkr.ecr.us-east-2.amazonaws.com/xray-daemon",
"cpu": 32,
"memoryReservation": 256,
"portMappings" : [
{
"hostPort": 0,
"containerPort": 2000,
"protocol": "udp"
}
],
},
{
"name": "scorekeep-api",
"image": "123456789012.dkr.ecr.us-east-2.amazonaws.com/scorekeep-api",
"cpu": 192,
"memoryReservation": 512,
"environment": [
{ "name" : "AWS_REGION", "value" : "us-east-2" },
{ "name" : "NOTIFICATION_TOPIC", "value" : "arn:aws:sns:us-east-2:123456789012:scorekeep-notifications" },
{ "name" : "AWS_XRAY_DAEMON_ADDRESS", "value" : "xray-daemon:2000" }
],
"portMappings" : [
{
"hostPort": 5000,
"containerPort": 5000
}
],
"links": [
"xray-daemon"
]
}
```