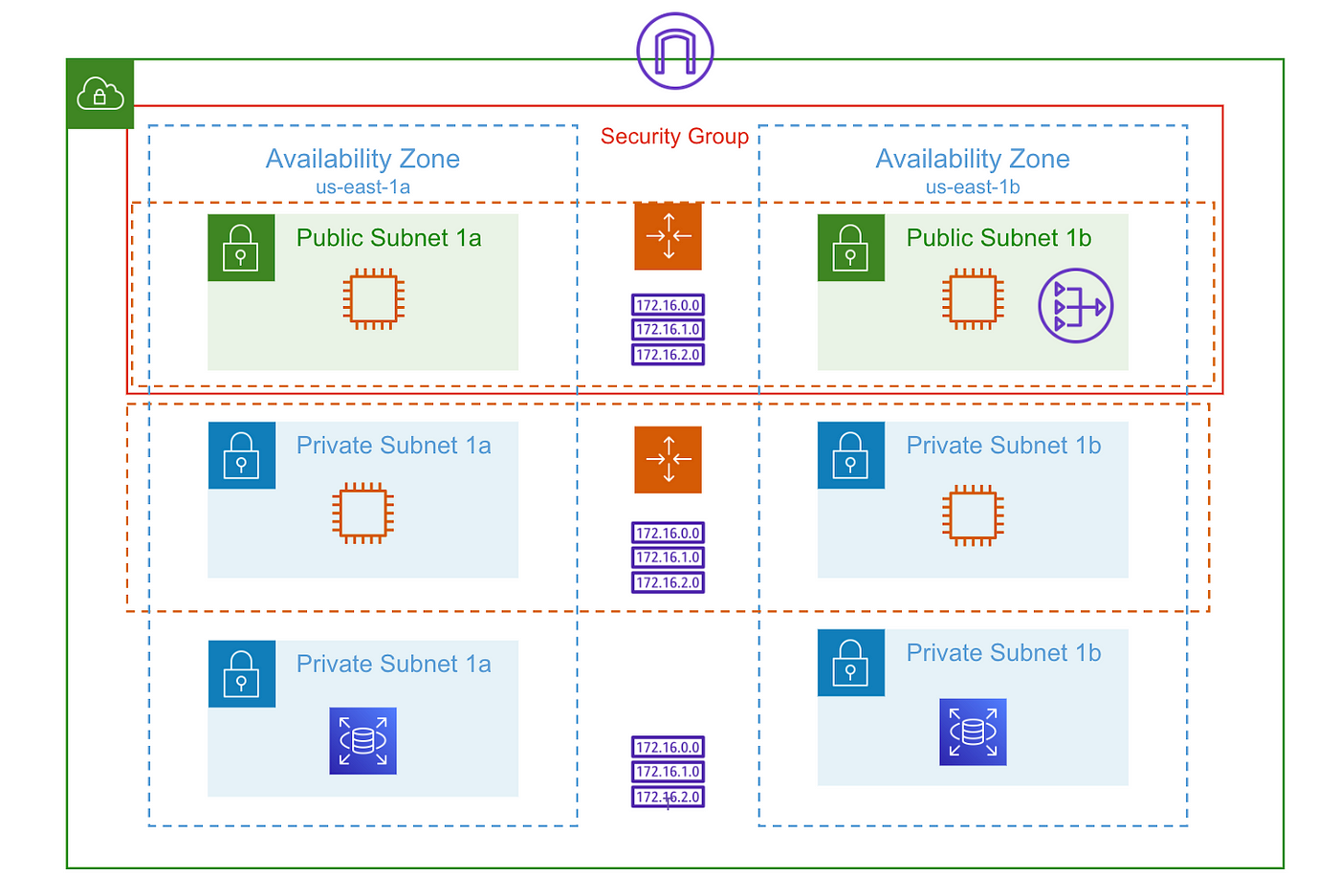
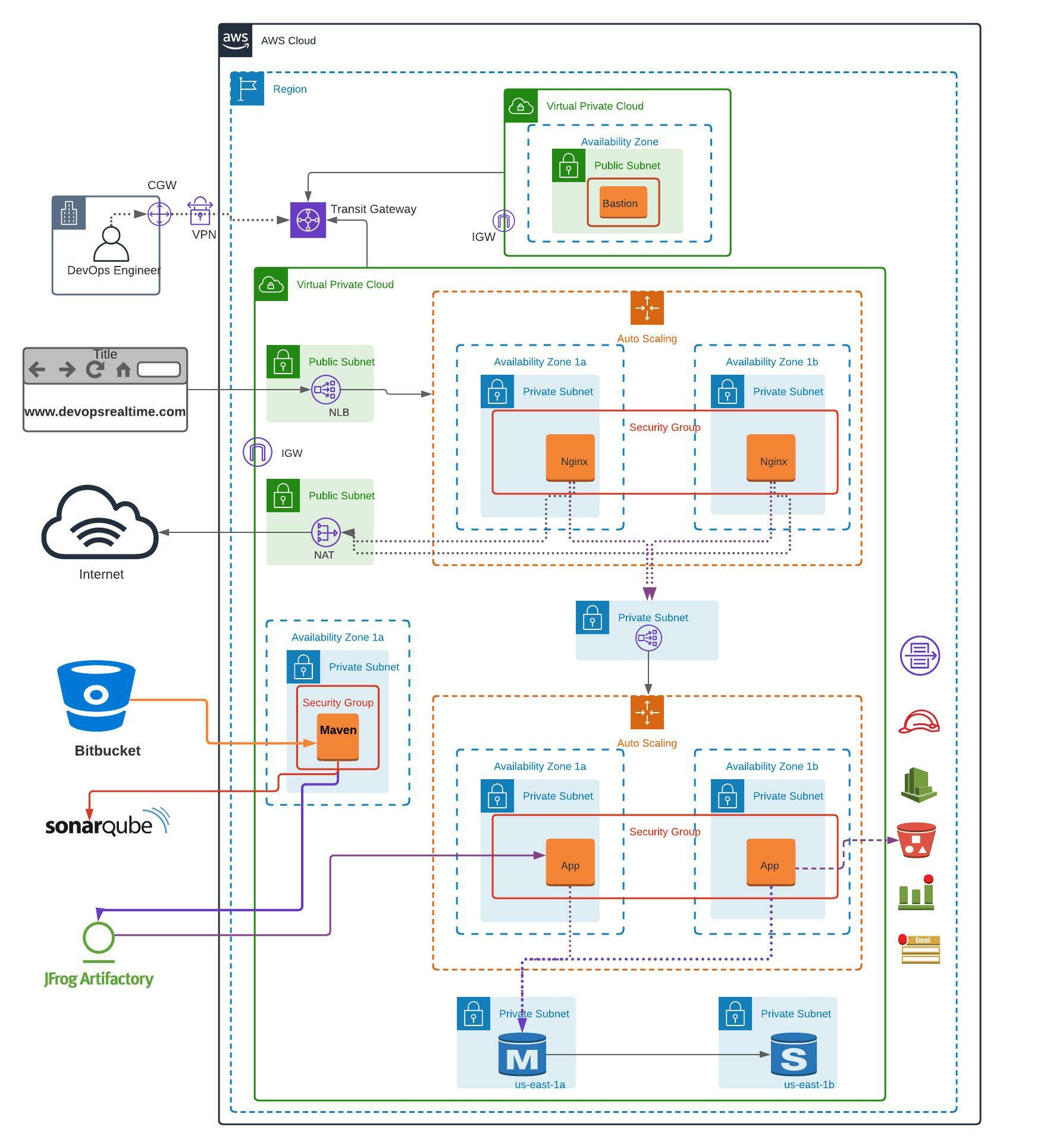
**Deploy Java Application on AWS 3-Tier Architecture**

[](https://camo.githubusercontent.com/efac0e6b1acf71928925095d91b53d2086334468f79d51c31608e2a45a294729/68747470733a2f2f696d6775722e636f6d2f623969487756632e706e67)

**TABLE OF CONTENTS**

1. [Goal](https://github.com/NotHarshhaa/DevOps-Projects/blob/master/DevOps%20Project-01/Project-01.md#goal)
2. [Pre-Requisites](https://github.com/NotHarshhaa/DevOps-Projects/blob/master/DevOps%20Project-01/Project-01.md#pre-requisites)
3. [Pre-Deployment](https://github.com/NotHarshhaa/DevOps-Projects/blob/master/DevOps%20Project-01/Project-01.md#pre-deployment)
4. [VPC Deployment](https://github.com/NotHarshhaa/DevOps-Projects/blob/master/DevOps%20Project-01/Project-01.md#vpc-deployment)
5. [Maven (Build)](https://github.com/NotHarshhaa/DevOps-Projects/blob/master/DevOps%20Project-01/Project-01.md#maven-build)
6. [3-Tier Architecture](https://github.com/NotHarshhaa/DevOps-Projects/blob/master/DevOps%20Project-01/Project-01.md#3-tier-architecture)
7. [Application Deployment](https://github.com/NotHarshhaa/DevOps-Projects/blob/master/DevOps%20Project-01/Project-01.md#application-deployment)
8. [Post-Deployment](https://github.com/NotHarshhaa/DevOps-Projects/blob/master/DevOps%20Project-01/Project-01.md#post-deployment)
9. [Validation](https://github.com/NotHarshhaa/DevOps-Projects/blob/master/DevOps%20Project-01/Project-01.md#validation)

**[](https://camo.githubusercontent.com/af8940ea2845ae5a70f794bc64888a988bf0c946a0e318d7aaac92279cd11f58/68747470733a2f2f696d6775722e636f6d2f33584630746c4a2e706e67)**

**Goal**

Goal of this project is to deploy scalable, highly available and secured Java application on 3-tier architecture and provide application access to the end users from public internet.

**Pre-Requisites**

1. Create AWS Free Tier account
2. Create GitHub account and create repository to keep this Java [Source Code](https://github.com/NotHarshhaa/DevOps-Projects/blob/master/DevOps%20Project-01/Java-Login-App)
3. Migrate Java Source Code to your own GitHub repository
4. Create account in Sonarcloud.
5. Create account in Jfrog cloud.

**Pre-Deployment**

1. Create Global AMI
   1. AWS CLI
   2. Cloudwatch agent
   3. Install AWS SSM agent
2. Create Golden AMI using Global AMI for Nginx application
   1. Install Nginx
   2. Push custom memory metrics to Cloudwatch.
3. Create Golden AMI using Global AMI for Apache Tomcat application
   1. Install Apache Tomcat
   2. Configure Tomcat as Systemd service
   3. Install JDK 11
   4. Push custom memory metrics to Cloudwatch.
4. Create Golden AMI using Global AMI for Apache Maven Build Tool
   1. Install Apache Maven
   2. Install Git
   3. Install JDK 11
   4. Update Maven Home to the system PATH environment variable

**VPC Deployment**

Deploy AWS Infrastructure resources as shown in the above architecture.

**VPC (Network Setup)**

1. Build VPC network ( 192.168.0.0/16 ) for Bastion Host deployment as per the architecture shown above.
2. Build VPC network ( 172.32.0.0/16 ) for deploying Highly Available and Auto Scalable application servers as per the architecture shown above.
3. Create NAT Gateway in Public Subnet and update Private Subnet associated Route Table accordingly to route the default traffic to NAT for outbound internet connection.
4. Create Transit Gateway and associate both VPCs to the Transit Gateway for private communication.
5. Create Internet Gateway for each VPC and update Public Subnet associated Route Table accordingly to route the default traffic to IGW for inbound/outbound internet connection.

**Bastion**

1. Deploy Bastion Host in the Public Subnet with EIP associated.
2. Create Security Group allowing port 22 from public internet

**Maven (Build)**

1. Create EC2 instance using Maven Golden AMI
2. Clone GitHub repository to VSCode and update the pom.xml with Sonar and JFROG deployment details.
3. Add settings.xml file to the root folder of the repository with the JFROG credentials and JFROG repo to resolve the dependencies.
4. Update application.properties file with JDBC connection string to authenticate with MySQL.
5. Push the code changes to feature branch of GitHub repository
6. Raise Pull Request to approve the PR and Merge the changes to Master branch.
7. Login to EC2 instance and clone the GitHub repository
8. Build the source code using maven arguments “-s settings.xml”
9. Integrate Maven build with Sonar Cloud and generate analysis dashboard with default Quality Gate profile.

**3-Tier Architecture**

**Database (RDS)**

1. Deploy Multi-AZ MySQL RDS instance into private subnets
2. Create Security Group allowing port 3306 from App instances and from Bastion Host.

**Tomcat (Backend)**

1. Create private facing Network Load Balancer and Target Group.
2. Create Launch Configuration with below configuration.
   1. Tomcat Golden AMI
   2. User Data to deploy .war artifact from JFROG into webapps folder.
   3. Security Group allowing Port 22 from Bastion Host and Port 8080 from private NLB.
3. Create Auto Scaling Group

**Nginx (Frontend)**

1. Create public facing Network Load Balancer and Target Group.
2. Create Launch Configuration with below configuration
   1. Nginx Golden AMI
   2. User Data to update proxy\_pass rules in nginx.conf file and reload nginx service.
   3. Security Group allowing Port 22 from Bastion Host and Port 80 from Public NLB.
3. Create Auto Scaling Group

**Application Deployment**

1. Artifact deployment taken care by User Data script during Application tier EC2 instance launch process.
2. Login to MySQL database from Application Server using MySQL CLI client and create database and table schema to store the user login data (Instructions are update in README.md file in the GitHub repo)

**Post-Deployment**

1. Configure Cronjob to push the Tomcat Application log data to S3 bucket and also rotate the log data to remove the log data on the server after the data pushed to S3 Bucket.
2. Configure Cloudwatch alarms to send E-Mail notification when database connections are more than 100 threshold.

**Validation**

1. Verify you as an administrator able to login to EC2 instances from session manager & from Bastion Host.
2. Verify if you as an end user able to access application from public internet browser.