

DEPLOYMENT OF A CONTAINERIZED JAVA BASED WEB APPLICATION USING JENKINS.

BY
EBUKA ONYIA

Introduction

In this project, we deployed a Java application in a containerized environment that is highly secured, available and scalable. In this project, we used docker as our container of choice to deploy our application in both staging and production environment, we also incorporated some security and monitoring measures for our infrastructure and application to make sure they are all in good condition to perform optimally. In addition, we automated a custom solution that would continuously deploy our application when even any change is made.

AIM: The aim of this project is to successfully deploy a java application in a containerized environment that highly available, highly secured and highly scalable.

Tech Stack:

- Terraform: Used for writing the IAC (infrastructure as a code)
- Java: used for writing the application
- AWS cloud: Cloud provider for the IAC
- Bash script: used for configuration of servers.
- Sonarqube: for scanning application code
- Nexus: storing artefact and docker image
- Owasp: dependency scanning tool.

SOURCE CODE LINK

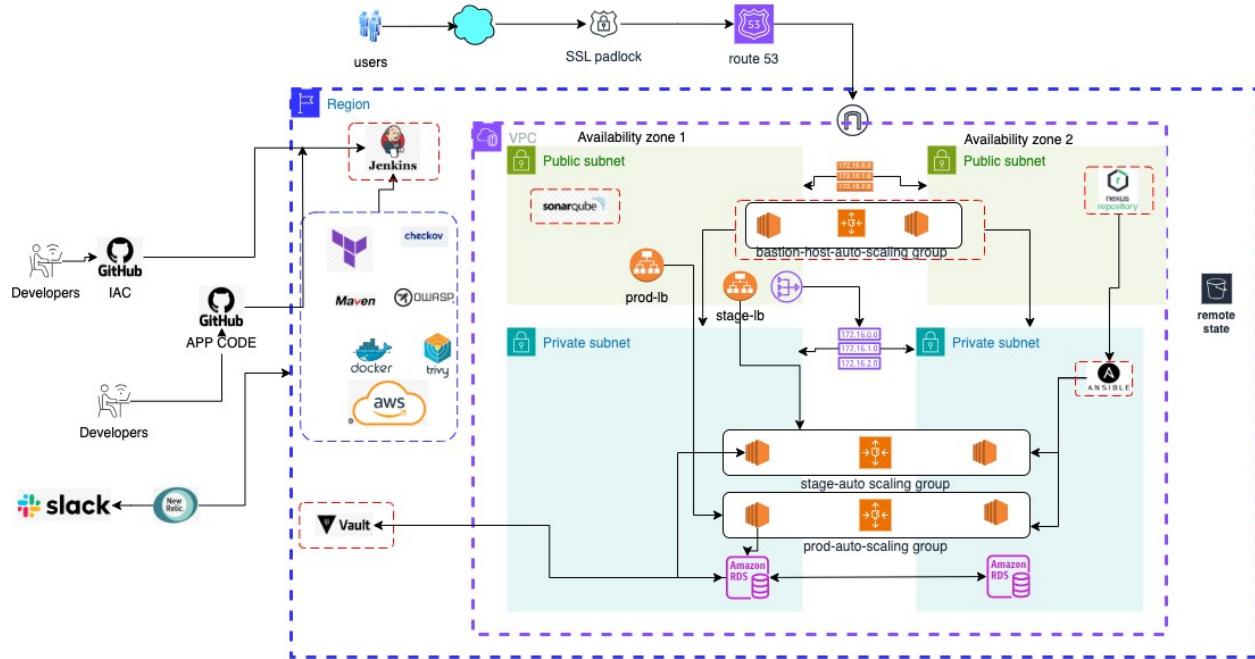
<https://github.com/onyiafranklin/autodiscovery-proj> infrastructure code link

<https://github.com/onyiafranklin/usteam-2> application source code(branch name >>eu-team2-app)

REQUIREMENTS

1. AWS Account
2. Domain name
3. VS CODE(code editor)
4. Github Account

ARCHITECTURAL DIAGRAM



The Architectural diagram above shows all networking components and services combined to successfully deploy our containerized java application using Jenkins pipeline. The following components were used.

1. **AWS Region:** This is a geographical area where AWS hosts its datacentres, and each region consist of multiple availability zones (ie isolated locations within a region). Here, we made use of EU-West-1 region for this project.

2. **VPC:** This is a logical isolated virtual network that we used to launch all the networking components for this project.
 3. **Availability Zone:** This is a physical separate datacentre within an AWS region, it has its own cooling, networking and power components that is isolated from failures in other Availability zones. This means that each AZ fails on its own without affecting the other availability zones. In this project, we made use of EU-west-1a and EU-west-1b.
 4. **Private subnets:** This is a network segment without a direct route to the internet. It can't be accessed via the internet and outbound access requires Nat-gateway. In this project, we made use of 2 private subnets on different availability zones to increase redundancy.
 5. **Public Subnets:** This is a network segment with access to the internet through the internet gateway. In this project we used 2 public subnets with provisioned on 2 different availability zones.
 6. **Internet-gateway:** This is an AWS service that allows communication between your VPC and internet. In this project, we are using it to route traffic from the public subnet to the internet.
 7. **Nat-gateway:** This is a managed AWS service that enables instances in private subnets to initiate outbound internet connections while blocking inbound traffic from the internet
 8. **Public route table:** A public route table is a critical networking component that controls traffic routing for public subnets in your VPC.
 9. **private route table** is a set of rules (routes) that determine how network traffic is directed within the private subnets of a VPC
-
10. **RDS database:** This is an aws managed service used to persist users data. We have deployed a multi AZ RDS database for high availability to create room for failover
 11. **Stage and production servers:** These are AWS ec2 instances where docker containers are installed that is used to launch the application. Stage servers are servers where user acceptability test, load performance and smoke test, after which application is deployed on prod servers.
 12. **Stage and Prod Load Balancers:** AWS ALBs/NLBs distributing traffic across stage and production environments, ensuring high availability and scalability
 13. **Bastion Host:** A secure jump server in a public subnet for SSH access to private instances.
 14. **Nexus Server:** Private artifact repository (e.g., Docker, Maven) hosted on EC2 or containers, In this project, we used the nexus to store artefact created by maven and docker image. here we created two registries one for maven hosted repo and the other for docker hosted repo.
 15. **SonarQube Servers:** Code quality analysis tools deployed on EC2/containers, integrated into CI/CD pipelines for static code scanning.
 16. **Vault Server:** This is HashiCorp Vault for secrets management (API keys, certificates) in private subnets with IAM integration. In this project, we used vault to store the rds credentials (username and password of our database). It was integrated with AWS KMS service which helped us to store the secrets of the vault that would help unseal the vault.
 17. **S3 Bucket (Remote State):** Secured bucket storing Terraform state files with versioning and encryption for IaC management.

18. **Jenkins Server:** CI/CD orchestration server with tools like **Trivy** (vulnerability scanning), **Maven** (Java builds), **Docker** (containerization), **OWASP** (security testing), and **Checkov** (IaC scanning).
19. **Slack:** Notification hub for alerts (e.g., Jenkins builds, security scans) via webhooks or bots .
20. **New Relic:** APM tool monitoring application performance, errors, and infrastructure metrics in prod/stage.
21. **Route53:** Managed DNS service routing traffic to ALBs/NLBs with health checks for failover.
22. **SSL Padlock/AWS ACM:** TLS certificates from ACM enabling HTTPS on ALBs/NLBs, auto renewed and integrated with Route53.
23. **GitHub:** Version control for code repositories, triggering Jenkins pipelines via webhooks for CI/CD automation.

PROCESS FLOW ON INFRASTRUCTURE PROVISIONNING AND APPLICATION DEPLOYMENT

Developers after development, push their code to github, which triggers the Jenkins pipeline to deploy the IAC code and application code.

First, before deployment of the IAC code, the Jenkins server and vault server are first provisioned. This is because we Jenkins server would be used to deploy every other infrastructure where our application would be sitting on, this IAC provisioning by Jenkins represents our First pipeline. Also, the vault server was also provisioned alongside with the Jenkins server, this is because the database credential stored in this vault would be used by the database provisioned by Jenkins server during database provisioning.

After the Jenkins and Vault provisioning, the Jenkins server with the help of “JENKINSFILE” containing stages of events that would be executed which would help to provision all other infrastructure where our application would be hosted.

After Infrastructure provisioning by our Jenkins server, the second pipeline with the help of another our application Jenkinsfile houses various steps that would help us deploy our application in a containerized environment would now kick off towards deploying our application.

NETWORK /DATA FLOW DURING USER'S EXPERIENCE

When user visits our application website(www.edenboutique.space) our route53 resolves the url to an ip address. With this , the user gains access to the website, when users request for data from the website, users traffic is encrypted via **TLS/SSL** (using ACM certificate attached to ALB),and sends data through internet gateway to the load balancers with data still encrypted, whereby the loadbalancers sends down decrypted information down to our docker servers in private subnet hosting the application.in same manner, data is sent back to the user in same manner through the Nat-gateway for consumption.

STEP BY STEP PROCEDURE TO SUCCESSFULLY PROVISION OUR INFRASTRUCTURE, DEPLOY OUR APPLICATION, PERSIST DATA ON DATABASE AND MONITORING USING NEWRELIC.

The steps below would lead us on the procedure on how to deploy our application successfully starting from infrastructure provisioning using terraform on vscode using a modularized template for various infrastructure to writing Jenkins file that would deploy our infrastructure down to the application docker file where the application is been containerized to the Jenkins file that deploy the containerized application, then to the connection of our application to persist data on our RDS and monitoring procedure for our application and infrastructure using Newrelic. The steps are listed below

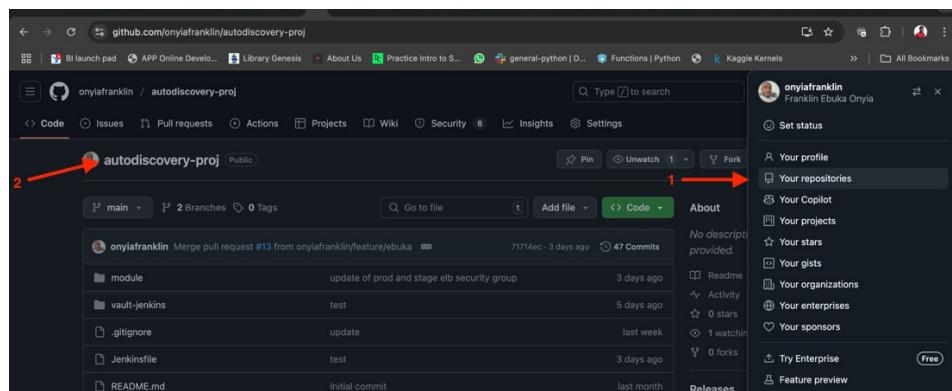
- a. Create a repo on Github and clone the repo on your local machine and setup file structure for all your code
- b. Create script that would create and destroy remote state bucket

- c. Vault and Jenkins server provisioning
- d. Modular code explanations for nexus ,sonarqube, database, stage environment, prod environment, database.
- e. Ansible code explanation
- f. Infrastructure jenkinsfile explanation
- g. Infrastructure provisioning setup for s3,vault and jenkins
- h. Infrastructure setup and provisioning using Jenkins pipeline
- i. Application jenkinsfile explanation
- j. Jenkins pipeline setup for application and application deployment
- k. Persisting our data to our database
- l. Change on application pipeline and redeployment to view change
- m. Monitoring using Newrelic

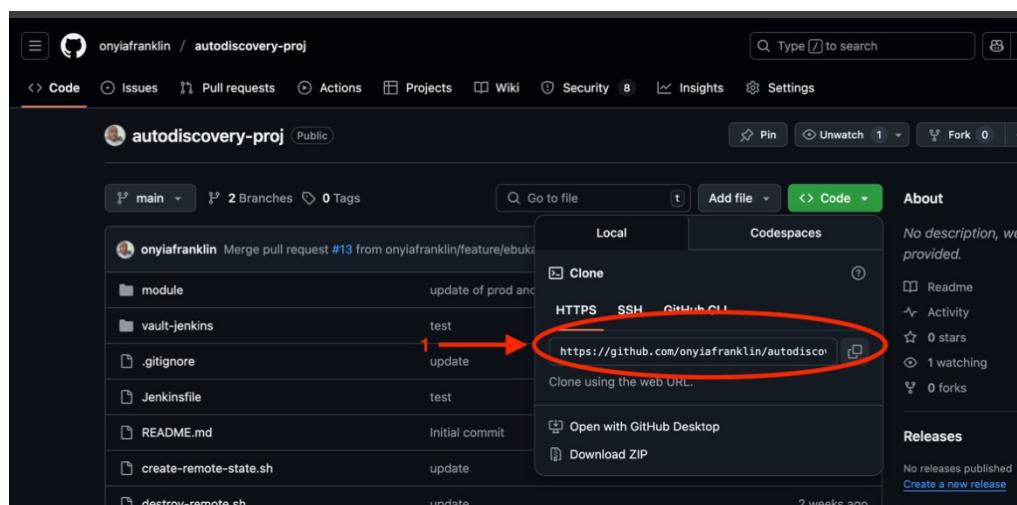
A. CREATE A REPOSITORY ON YOUR GITHUB AND CLONE ON YOUR LOCAL MACHINE

Here we are to create a repository account for collaboration and storing and working on different versions of code.

1. Go to www.github.com and create an account, there after create a repository.



2. Copy the url to clone the repository on your local machine, go to your terminal and type “git clone <url>”

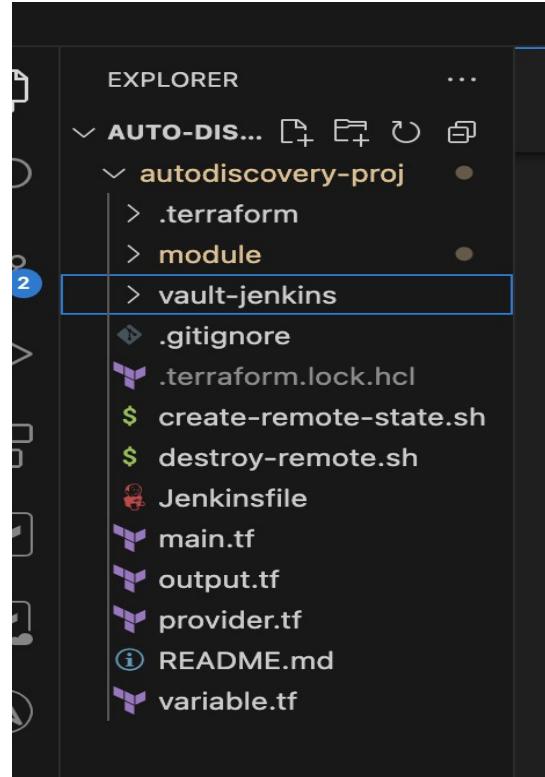


3. After cloning on your terminal, Create modules and file for the project

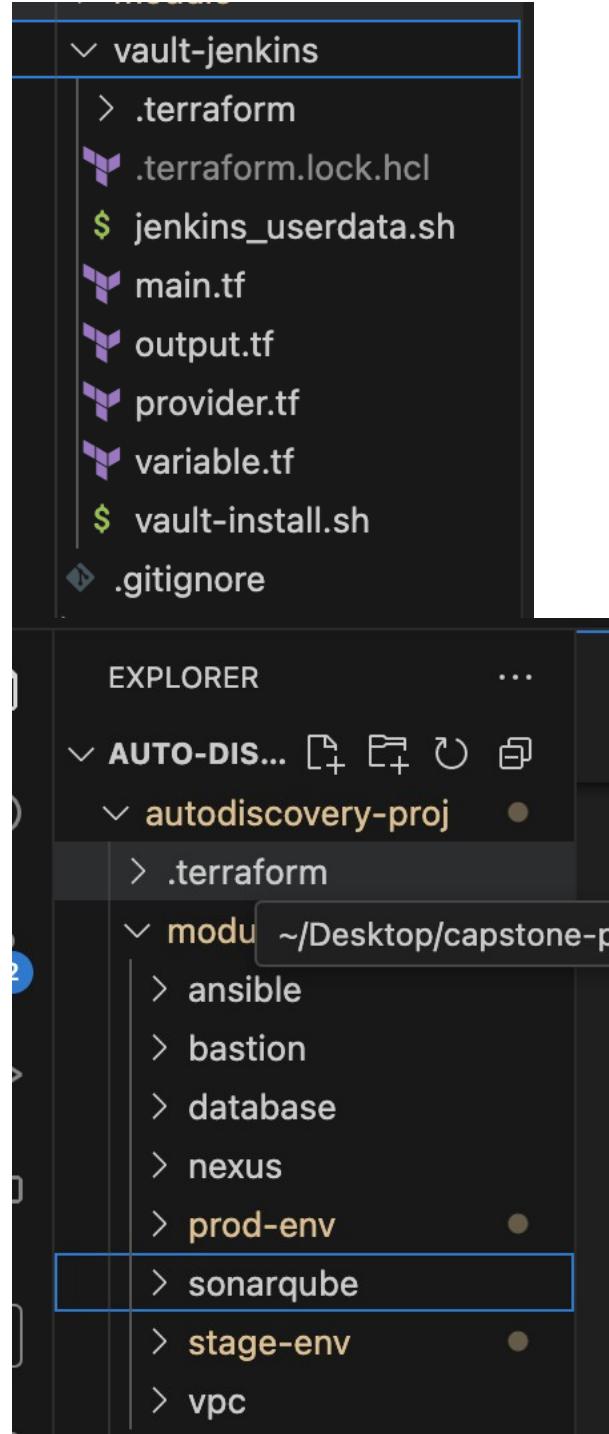
Create the folders and files show below in the cloned repository folder.this includes

the vault-jenkins folder where we would provision the vault and Jenkins server

Create the root main,output,variable .tf files



4. Create modules folder where we would provision all the modules for our sonarqube,nexus, prod ,stage server, and database modules.



B. CREATE SCRIPT THAT WOULD CREATE AND DESTROY REMOTE STATE BUCKET.

Here we created a script called “create remote state” the main function of this script is to create an s3 bucket that would be used to store our terraform state file. This is because you won’t be the only one working on this project, you have other team members working along with you, it helps to preserver the state of your terraform code to prevent simultaneous change at same time when both teams are working on the code. So therefore, it locks the state when the first member of the team is working on the project and when the first person completes its work, the second team member can now pull the latest terraform state file and works it as well.

CREATE S3 BUCKET

The code snippet helps us to create bucket called “bucket-pet-adoption” in eu-west-2 region. The final part of the code also helps us to create Jenkins and vault server.

```
#!/bin/bash
set -euo pipefail # Enable strict error handling

# Set Variables
BUCKET_NAME="bucket-pet-adoption"
AWS_REGION="eu-west-2"
PROFILE="bukky_int"

# Function to handle errors
handle_error() {
    echo "⚠ Error: $1"
    exit 1
}

# Create S3 Bucket
echo "⚠ Creating S3 bucket: $BUCKET_NAME..."
if aws s3api head-bucket --bucket "$BUCKET_NAME" --region "$AWS_REGION" --profile "$PROFILE" 2>/dev/null;
then
    echo "⚠ Bucket '$BUCKET_NAME' already exists. Skipping creation."
else
    if ! aws s3api create-bucket --bucket "$BUCKET_NAME" --region "$AWS_REGION" --profile "$PROFILE" --create-bucket-configuration LocationConstraint="$AWS_REGION"; then
        handle_error "Failed to create S3 bucket '$BUCKET_NAME'."
    fi
    echo "⚠ S3 bucket '$BUCKET_NAME' created successfully."
fi

# # Enable versioning
```

```

echo "Enabling versioning for S3 bucket..."
if ! aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-configuration Status=Enabled --region "$AWS_REGION" --profile "$PROFILE"; then
    handle_error "Failed to enable versioning for S3 bucket '$BUCKET_NAME'.""
fi
echo "Versioning enabled successfully.

echo "S3 Remote State Management Setup Complete!"
echo "S3 Bucket: $BUCKET_NAME"

# provision the vault and jenkins server
echo "Provisioning Vault and Jenkins server..."
cd ./vault-jenkins
terraform init
terraform fmt --recursive
terraform validate
terraform apply -auto-approve
terraform output
echo "Vault and Jenkins server provisioned successfully."

```

DESTROY S3 BUCKET

This the major role of this script is to destroy the s3 bucket created as well as destroy the jenkins and vault server

```

#!/bin/bash
set -eou pipefail # Enable strict error handling

# Set Variables
BUCKET_NAME="bucket-pet-adoption"
AWS_REGION="eu-west-2"
PROFILE="bukky_int"

# destroy vault and jenkins server
cd ./vault-jenkins
terraform destroy -auto-approve
terraform output

# Function to handle errors
handle_error() {
    echo "Error: $1"
    exit 1
}

# Ensure AWS CLI is installed
if ! command -v aws >/dev/null; then
    handle_error "AWS CLI is not installed. Please install it and retry."
fi

```

```

# Check if S3 bucket exists before attempting deletion
if aws s3api head-bucket --bucket "$BUCKET_NAME" --region "$AWS_REGION" --
profile "$PROFILE" 2>/dev/null; then
    echo "☒ Deleting all objects from S3 bucket: $BUCKET_NAME..."

    # Delete all versions of objects
    versions=$(aws s3api list-object-versions --bucket "$BUCKET_NAME" --
query='{Objects: Versions[].{Key:Key,VersionId:VersionId}}' --output=json --
profile "$PROFILE")
    if [ "$versions" != '{"Objects":[]}' ]; then
        echo "$versions" > delete.json
        if ! aws s3api delete-objects --bucket "$BUCKET_NAME" --delete
file://delete.json --profile "$PROFILE"; then
            handle_error "Failed to delete versioned objects from S3 bucket
'$BUCKET_NAME'."
        fi
        rm -f delete.json
    fi

    # Delete all delete markers
    delete_markers=$(aws s3api list-object-versions --bucket "$BUCKET_NAME" --
--query='{Objects: DeleteMarkers[].{Key:Key,VersionId:VersionId}}' --
output=json --profile "$PROFILE")
    if [ "$delete_markers" != '{"Objects":[]}' ]; then
        echo "$delete_markers" > delete_markers.json
        if ! aws s3api delete-objects --bucket "$BUCKET_NAME" --delete
file://delete_markers.json --profile "$PROFILE"; then
            handle_error "Failed to delete delete markers from S3 bucket
'$BUCKET_NAME'."
        fi
        rm -f delete_markers.json
    fi

    echo "☒ All objects, including versioned ones, deleted from S3 bucket."

    echo "☒ Deleting S3 bucket: $BUCKET_NAME..."
    if ! aws s3api delete-bucket --bucket "$BUCKET_NAME" --region
"$AWS_REGION" --profile "$PROFILE"; then
        handle_error "Failed to delete S3 bucket '$BUCKET_NAME'."
    fi
    echo "☒ S3 bucket '$BUCKET_NAME' deleted successfully."
else
    echo "⚠ S3 bucket '$BUCKET_NAME' does not exist. Skipping deletion."
fi

echo "☒ S3 Bucket Cleanup Complete!"

```

C. VAULT AND JENKINS SERVER PROVISIONING

The vault and Jenkins server are provisioned immediately after the bucket is created. This is because the Jenkins server needs to be up and running because it would be used to provision the infrastructure where our application would be running. Also the vault server needs to be up and running because the database that would be provisioned by the Jenkins needs to make an API call to the vault server to use the credentials stored in the vault server.

For vault

The resource block "resource "aws_kms_key" "kms-key" below shows the creation of KMS key that would be used to manage the vault secrets.

This followed by creation of IAM role and policy using the "resource "aws_iam_role" "vault_role" and resource "aws_iam_role_policy" "vault_kms_access" resource block "that would enable the vault server to perform some encryption and decryption role.

Next is the resource "aws_iam_instance_profile" "vault-profile" whose role be attached to the vault server during creation. This is followed by "resource "aws_security_group" "vault_sg"" which is opened at port 22 for ssh, port 80 for http and port 8200 is default port for hasicorp vault.

Also we have created a network /elastic load balancer resource "aws_elb" "elb-vault" for the vault server opened on port 443 so it can interface with the external environment and receive request, it was also opened on instance port 8200 to interact with the vault server itself.

Furthermore, we created a security group for the vault elastic load balancer resource "aws_security_group" "elb-vault-sg" with ingress at port 443 and egress at anyport. Vaultserver with resource block resource "aws_instance" "vault_server" is not left out as it was created in the default VPC here for testing purpose. Also, we created an AWS certificate manager resource resource "aws_acm_certificate" "acm-cert" for our domain and subdomain together with its record. after the certificate has been issued, we moved ahead with creation of records for our vault server on resource "aws_route53_record" "vault-record" .

For Jenkins,

This started with the creation of IAM role for Jenkins server resource "aws_iam_role" "jenkins-role" in which administrator role resource "aws_iam_role_policy_attachment" "jenkins-role-attachment" is attached to the role, after which iam instance profile (resource "aws_iam_instance_profile" "jenkins-profile") is attached to the Jenkins server during creation

Next is the creation of Security group(resource "aws_security_group" "jenkins_sg") for Jenkins server which is opened on opened at port 22 for ssh, port 443 for https and port 8080 is default port for Jenkins

Also we have created a network /elasticload balancer resource (resource "aws_elb" "elb_jenkins") for the Jenkins server opened on port 443 so it can interface with the external environment and

receive request,it was also opened on instance port 8080 to interact with the jenkins server itself. Furthermore, we created a security group for the jenkins elastic load balancer resource (resource "aws_security_group" "jenkins-elb-sg") with ingress at port 443 and egress at anyport. Jenkins server with resource block resource resource ("aws_instance" "jenkins-server") is not left out as it was created in the default VPC here for testing purpose. Also, we created an AWS certificate manager resource resource "aws_acm_certificate" "acm-cert" for our domain and subdomain together with its record. after the certificate has been issued, we moved ahead with creation of records for our jenkins server on resource (resource "aws_route53_record" "jenkins-record").

```
main.tf
locals {
    name = "auto-discov"
}
#Creating kms key
resource "aws_kms_key" "kms-key" {
    description          = "${local.name}-vault-kms-key" // the kms key
    deletion_window_in_days = 30                         // 30 days
    enable_key_rotation   = true                          // true
}
# alias of the kms key
resource "aws_kms_alias" "kms-key" {
    name      = "alias/${local.name}-kms-key"
    target_key_id = aws_kms_key.kms-key.key_id
}

resource "aws_iam_role" "vault_role" {
    name = "${local.name}-vault-role"

    assume_role_policy = jsonencode({
        Version = "2012-10-17",
        Statement = [
            {
                Effect = "Allow",
                Principal = {
                    Service = "ec2.amazonaws.com"
                },
                Action = "sts:AssumeRole"
            }
        ]
    })
}

resource "aws_iam_role_policy" "vault_kms_access" {
    name = "${local.name}-vault_kms-access"
    role = aws_iam_role.vault_role.id

    policy = jsonencode({
        Version = "2012-10-17",
        Statement = [
            {
                Effect = "Allow",

```

```

        Action = [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
        ],
        Resource = "${aws_kms_key.kms-key.arn}"
    }
]
})
}
resource "aws_iam_instance_profile" "vault-profile" {
    name = "${local.name}-vault-profile"
    role = aws_iam_role.vault_role.name
}

# Create Vault Security Group
resource "aws_security_group" "vault_sg" {
    name      = "vault-sg"
    description = "Allow SSH, HTTP, HTTPS, and Vault UI/API"

    ingress {
        from_port   = 22
        to_port     = 22
        protocol    = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }

    ingress {
        from_port   = 80
        to_port     = 80
        protocol    = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
    ingress {
        from_port   = 8200
        to_port     = 8200
        protocol    = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }

    ingress {
        from_port   = 443
        to_port     = 443
        protocol    = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }

    egress {
        from_port   = 0
        to_port     = 0
        protocol    = "-1"
        cidr_blocks = ["0.0.0.0/0"]
    }
}
# Ubuntu AMI lookup

```

```

data "aws_ami" "ubuntu" {
  most_recent = true
  owners      = ["099720109477"] # Canonical
  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-*"]
  }
  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }
}
resource "aws_instance" "vault_server" {
  ami           = data.aws_ami.ubuntu.id # Ubuntu in eu-west-2
  instance_type = "t2.medium"
  key_name      = aws_key_pair.public_key.key_name
  security_groups = [aws_security_group.vault_sg.name]
  iam_instance_profile = aws_iam_instance_profile.vault-profile.id
  user_data = templatefile("./vault-install.sh", {
    var1 = "eu-west-2",
    var2 = aws_kms_key.kms-key.id
  })

  tags = {
    Name = "${local.name}-VaultServer"
  }
}

# Create keypair resource
resource "tls_private_key" "keypair" {
  algorithm = "RSA"
  rsa_bits  = 4096
}
resource "local_file" "private_key" {
  content      = tls_private_key.keypair.private_key_pem
  filename     = "${local.name}-key.pem"
  file_permission = "400"
}
resource "aws_key_pair" "public_key" {
  key_name   = "${local.name}-key"
  public_key = tls_private_key.keypair.public_key_openssh
}
# Data source to get the latest RedHat AMI
data "aws_ami" "redhat" {
  most_recent = true
  owners      = ["309956199498"] # RedHat's owner ID
  filter {
    name   = "name"
    values = ["RHEL-9*"]
  }
  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }
  filter {
    name   = "architecture"
    values = ["x86_64"]
  }
}

```

```

        }
    }
resource "aws_instance" "jenkins-server" {
    ami                      = data.aws_ami.redhat.id # redhat in eu-west-2
    instance_type             = "t2.medium"
    key_name                 = aws_key_pair.public_key.id
    associate_public_ip_address = true
    vpc_security_group_ids   = [aws_security_group.jenkins_sg.id]
    iam_instance_profile     = aws_iam_instance_profile.jenkins-profile.id
    root_block_device {
        volume_size = 30      # Size in GB
        volume_type = "gp3"   # General Purpose SSD (recommended)
        encrypted   = true    # Enable encryption (best practice)
    }
    user_data = templatefile("./jenkins_userdata.sh", {
        nr-key      = "NRAK-4FNJBSGOTULJ4XCZW4P2J0MPOKY",
        nr-acc-id   = 6496342
    })
    tags = {
        Name = "${local.name}-jenkins-server"
    }
}
# Create IAM role for Jenkins
resource "aws_iam_role" "jenkins-role" {
    name = "${local.name}-jenkins-role"

    assume_role_policy = jsonencode({
        Version = "2012-10-17",
        Statement = [
            {
                Effect = "Allow",
                Principal = {
                    Service = "ec2.amazonaws.com"
                },
                Action = "sts:AssumeRole"
            }
        ]
    })
}
# Attach the policy to the role
resource "aws_iam_role_policy_attachment" "jenkins-role-attachment" {
    role      = aws_iam_role.jenkins-role.name
    policy_arn = "arn:aws:iam::aws:policy/AdministratorAccess"
}
# Attach the policy to the role
resource "aws_iam_instance_profile" "jenkins-profile" {
    name = "${local.name}-jenkins-profile"
    role = aws_iam_role.jenkins-role.name
}

# Create jenkins security group
resource "aws_security_group" "jenkins_sg" {
    name          = "${local.name}-jenkins-sg"
    description   = "Allow SSH and HTTPS"
    ingress {

```

```

        from_port    = 22
        to_port     = 22
        protocol    = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }

ingress {
    from_port    = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
}

ingress {
    from_port    = 8080
    to_port     = 8080
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
}
egress {
    from_port    = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
}
}

#create a time sleep resource that allow terraform to wait till vault server
is ready
resource "time_sleep" "wait_3_min" {
    depends_on      = [aws_instance.vault_server]
    create_duration = "180s"
}

#create null resource to fetch vault token
resource "null_resource" "fetch_token" {
    depends_on = [time_sleep.wait_3_min]
    # create terraform provisioner to help fetch token file from the vault
server
    provisioner "local-exec" {
        command = "scp -o StrictHostKeyChecking=no -i ./${local.name}-key.pem
ubuntu@${aws_instance.vault_server.public_ip}:/home/ubuntu/token.txt ."
    }

    # provisioner "locai-exec" {
    #     interpreter=["bash", "-c"]
    #     command= "sed -i '' \"s/token = \\\\\".*\\\\\"/token = \\\\\"$(cat
./token.txt)\\\\\"/\" ..//provider.t"
    # }
    provisioner "local-exec" {
        when      = destroy
        command = "rm -f ./token.txt"
    }
}
# Create ACM certificate with DNS validation
resource "aws_acm_certificate" "acm-cert" {
    domain_name          = var.domain
}

```

```

subject_alternative_names = ["*.${var.domain}"]
validation_method          = "DNS"
lifecycle {
  create_before_destroy = true
}

tags = {
  Name = "${local.name}-acm-cert"
}
}

data "aws_route53_zone" "acp-zone" {
  name      = var.domain
  private_zone = false
}

# Fetch DNS Validation Records for ACM Certificate
resource "aws_route53_record" "acm_validation_record" {
  for_each = {
    for dvo in aws_acm_certificate.acm-cert.domain_validation_options :
    dvo.domain_name => {
      name      = dvo.resource_record_name
      record   = dvo.resource_record_value
      type     = dvo.resource_record_type
    }
  }

  # Create DNS Validation Record for ACM Certificate
  zone_id      = data.aws_route53_zone.acp-zone.zone_id
  allow_overwrite = true
  name        = each.value.name
  type        = each.value.type
  ttl         = 60
  records     = [each.value.record]
  depends_on   = [aws_acm_certificate.acm-cert]
}

# Validate the ACM Certificate after DNS Record Creation
resource "aws_acm_certificate_validation" "team2_cert_validation" {
  certificate_arn      = aws_acm_certificate.acm-cert.arn
  validation_record_fqdns = [for record in
  aws_route53_record.acm_validation_record : record.fqdn]
  depends_on            = [aws_acm_certificate.acm-cert]
}

#Create Security Group for Vault Elastic Load Balancer
resource "aws_security_group" "elb-vault-sg" {
  name      = "elb-vault-sg"
  description = "Allow HTTPS"

  ingress {
    from_port    = 443
    to_port      = 443
    protocol     = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

```

```

    egress {
      from_port   = 0
      to_port     = 0
      protocol    = "-1"
      cidr_blocks = ["0.0.0.0/0"]
    }
  }

# Create load balancer for Vault Server
resource "aws_elb" "elb-vault" {
  name          = "vault-elb"
  availability_zones = ["eu-west-2a", "eu-west-2b"]

  listener {
    instance_port      = 8200
    instance_protocol  = "http"
    lb_port            = 443
    lb_protocol        = "https"
    ssl_certificate_id = aws_acm_certificate.acm-cert.arn
  }

  health_check {
    healthy_threshold  = 2
    unhealthy_threshold = 2
    timeout            = 3
    target              = "TCP:8200"
    interval            = 30
  }

  instances           = [aws_instance.vault_server.id]
  cross_zone_load_balancing = true
  idle_timeout        = 400
  connection_draining = true
  connection_draining_timeout = 400

  tags = {
    Name = "${local.name}-elb-vault"
  }
}

# Create Route 53 Hosted Zone
data "aws_route53_zone" "vault-zone" {
  name      = var.domain
  private_zone = false
}

# Create Route 53 A Record for Vault Server
resource "aws_route53_record" "vault-record" {
  zone_id = data.aws_route53_zone.vault-zone.zone_id
  name    = "vault.${var.domain}"
  type    = "A"
  alias {
    name          = aws_elb.elb-vault.dns_name
    zone_id       = aws_elb.elb-vault.zone_id
    evaluate_target_health = true
  }
}

```

```

# Create elastic Load Balancer for Jenkins
resource "aws_elb" "elb_jenkins" {
    name          = "elb-jenkins"
    security_groups = [aws_security_group.jenkins-elb-sg.id]
    availability_zones = ["eu-west-2a", "eu-west-2b"]
    listener {
        instance_port      = 8080
        instance_protocol  = "HTTP"
        lb_port            = 443
        lb_protocol        = "HTTPS"
        ssl_certificate_id = aws_acm_certificate.acm-cert.id
    }
    health_check {
        healthy_threshold   = 3
        unhealthy_threshold = 2
        interval           = 30
        timeout             = 5
        target              = "TCP:8080"
    }
    instances          = [aws_instance.jenkins-server.id]
    cross_zone_load_balancing = true
    idle_timeout       = 400
    connection_draining = true
    connection_draining_timeout = 400
    tags = {
        Name = "${local.name}-jenkins-server"
    }
}
# Create Security group for the jenkins elb
#
resource "aws_security_group" "jenkins-elb-sg" {
    name      = "${local.name}-jenkins-elb-sg"
    description = "Allow HTTPS"

    ingress {
        from_port  = 443
        to_port    = 443
        protocol   = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
    egress {
        from_port  = 0
        to_port    = 0
        protocol   = "-1"
        cidr_blocks = ["0.0.0.0/0"]
    }
}

# Create Route 53 record for jenkins server
resource "aws_route53_record" "jenkins-record" {
    zone_id = data.aws_route53_zone.acp-zone.zone_id
    name    = "jenkins.${var.domain}"
    type    = "A"
    alias {
        name          = aws_elb.elb_jenkins.dns_name
        zone_id       = aws_elb.elb_jenkins.zone_id
        evaluate_target_health = true
    }
}

```

```
}
```

userdata for Jenkins server

here we installed all dependencies for our jenkins server. some of them include maven,wget,git, jenkins, docker,trivy,awscli, newrelic

```
#!/bin/bash
sudo yum update -y
sudo yum install wget -y
sudo yum install maven -y
sudo yum install git pip -y
sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
sudo yum upgrade -y
sudo yum install java-17-openjdk -y
sudo yum install jenkins -y
sudo sed -i 's/^User=jenkins/User=root/' /usr/lib/systemd/system/jenkins.service
sudo systemctl daemon-reload
sudo systemctl start jenkins
sudo systemctl enable jenkins
sudo systemctl start jenkins
sudo usermod -aG jenkins ec2-user
# Install trivy for container scanning
RELEASE_VERSION=$(grep -Po '(?=<VERSION_ID=")[0-9]' /etc/os-release)
cat << EOT | sudo tee -a /etc/yum.repos.d/trivy.repo
[trivy]
name=Trivy repository
baseurl=https://aquasecurity.github.io/trivy-repo/rpm/releases/
$RELEASE_VERSION/\$basearch/
gpgcheck=0
enabled=1
EOT
sudo yum -y update
sudo yum -y install trivy
# installing docker
sudo yum install -y yum-utils
sudo yum config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo
sudo yum install docker-ce -y
sudo systemctl start docker
sudo systemctl enable docker

# Installing awscli
sudo yum install unzip -y
sudo curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
sudo unzip awscliv2.zip
sudo ./aws/install
sudo ln -svf /usr/local/bin/aws /usr/bin/aws

# install newrelic agent
```

```

curl -Ls
https://download.newrelic.com/install/newrelic-cli/scripts/install.sh | bash
&& sudo NEW_RELIC_API_KEY="${nr-key}" NEW_RELIC_ACCOUNT_ID="${nr-acc-id}"
NEW_RELIC_REGION=EU /usr/local/bin/newrelic install -y
sudo hostnamectl set-hostname jenkins

```

Userdata for vault server

here we intalled all dependencies such as the consul(database of the vault server), we also installed the vault and initialise the vault to get the secrects that would be used to unseal the vault. vault has 5 secrects stored init but we require only 3 to unseal the vault, so with the help of kms(key management service), we are able to manage these secrects in kms .

```

#!/bin/bash

# Update package repositories
sudo apt update

# Download and install Consul
sudo wget
https://releases.hashicorp.com/consul/1.7.3/consul_1.7.3_linux_amd64.zip
sudo apt install unzip -y
sudo unzip consul_1.7.3_linux_amd64.zip
sudo mv consul /usr/bin/

# Create a Consul systemd service
sudo cat <<EOT>> /etc/systemd/system/consul.service
[Unit]
Description=Consul
Documentation=https://www.consul.io/

[Service]
ExecStart=/usr/bin/consul agent -server -ui -data-dir=/temp/consul -
bootstrap-expect=1 -node=vault -bind=$(hostname -i)
-config-dir=/etc/consul.d/
ExecReload=/bin/kill -HUP $MAINPID
LimitNOFILE=65536

[Install]
WantedBy=multi-user.target
EOT

# Create Consul configuration directory and UI settings
sudo mkdir /etc/consul.d
sudo cat <<EOT>> /etc/consul.d/ui.json
{
    "addresses": {
        "http": "0.0.0.0"
    }
}
EOT

```

```

# Reload systemd, start, and enable Consul service
sudo systemctl daemon-reload
sudo systemctl start consul
sudo systemctl enable consul

# Download and install Vault
sudo apt update
sudo wget
https://releases.hashicorp.com/vault/1.5.0/vault_1.5.0_linux_amd64.zip
sudo unzip vault_1.5.0_linux_amd64.zip
sudo mv vault /usr/bin/

# Create Vault configuration file
sudo mkdir /etc/vault/
sudo cat <<EOT>> /etc/vault/config.hcl
storage "consul" {
    address = "127.0.0.1:8500"
    path     = "vault/"
}

listener "tcp" {
    address      = "0.0.0.0:8200"
    tls_disable   = 1
}

seal "awskms" {
    region      = "${var1}"
    kms_key_id  = "${var2}"
}
ui = true #to access/visualize vault thru the browser
EOT

# Create Vault systemd service
#creating a service file for vault
sudo cat <<EOT>> /etc/systemd/system/vault.service
[Unit]
Description=Vault
Documentation=https://www.vault.io/

[Service]
ExecStart=/usr/bin/vault server -config=/etc/vault/config.hcl
ExecReload=/bin/kill -HUP $MAINPID
LimitNOFILE=65536

[Install]
WantedBy=multi-user.target
EOT

# Reload systemd, start, and enable Vault service
sudo systemctl daemon-reload
export VAULT_ADDR="http://localhost:8200"
cat <<EOT > /etc/profile.d/vault.sh
export VAULT_ADDR="http://localhost:8200"
export VAULT_SKIP_VERIFY=true
EOT
vault -autocomplete-install

```

```

complete -C /usr/bin/vault vault

# Notify once provisioned
echo "Vault server provisioned successfully."
# Start Vault service
sudo systemctl start vault
sudo systemctl enable vault
sleep 20
# #Set vault token/secret username and password
touch /home/ubuntu/output.txt
vault operator init > /home/ubuntu/output.txt
grep -o 's\.[A-Za-z0-9]\{24\}' /home/ubuntu/output.txt >
/home/ubuntu/token.txt
token_content=$(
```

```
vault login $token_content
```

```
vault secrets enable -path=secret/ kv #directory to store secrets on the
```

```
vault server
vault kv put secret/database username=petclinic password=petclinic
# Set hostname to Vault
sudo hostnamectl set-hostname Vault
```

D. MODULAR CODE EXPLANATIONS FOR NEXUS ,SONARQUBE, DATABASE, STAGE ENVIRONMENT, PROD ENVIRONMENT, DATABASE.

VPC Module

This is a virtual private cloud set apart to host all networking components for our infrastructure. here we have deployed our custom VPC, interner gateway, private and public subnets,public and private route tables in ADDITION TO THE NATgateway. we also created a keypair that would be used during our server creation. this keypair would enable us access whenever we want to ssh into the server . This was created using the RSA algorithm

```

main.tf
#Creating the vpc
resource "aws_vpc" "vpc" {
  cidr_block      = "10.0.0.0/16"
  instance_tenancy = "default"

  tags = {
    Name = "${var.name}-vpc"
  }
}

# create public subnet 1
resource "aws_subnet" "pub_sub1" {
  vpc_id          = aws_vpc.vpc.id
  cidr_block      = "10.0.1.0/24"
  availability_zone = var.azi1

  tags = {
    Name = "${var.name}-pub_sub1"
  }
}
```

```

}

# create public subnet 2
resource "aws_subnet" "pub_sub2" {
  vpc_id          = aws_vpc.vpc.id
  cidr_block      = "10.0.2.0/24"
  availability_zone = var.az2

  tags = {
    Name = "${var.name}-pub_sub2"
  }
}

# create private subnet 1
resource "aws_subnet" "pri_sub1" {
  vpc_id          = aws_vpc.vpc.id
  cidr_block      = "10.0.3.0/24"
  availability_zone = var.az1

  tags = {
    Name = "${var.name}-pri_sub1"
  }
}

# create private subnet 2
resource "aws_subnet" "pri_sub2" {
  vpc_id          = aws_vpc.vpc.id
  cidr_block      = "10.0.4.0/24"
  availability_zone = var.az2

  tags = {
    Name = "${var.name}-pri_sub2"
  }
}

# create internet gateway
resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.vpc.id

  tags = {
    Name = "${var.name}-igw"
  }
}

# create elastic ip for NAT gateway
resource "aws_eip" "eip" {
  domain = "vpc"

  tags = {
    Name = "${var.name}-eip"
  }
}

# create NAT gateway
resource "aws_nat_gateway" "ngw" {
  allocation_id = aws_eip.eip.id
  subnet_id     = aws_subnet.pub_sub1.id
}

```

```

tags = {
    Name = "${var.name}-ngw"
}
depends_on = [aws_eip.eip] # Ensuring the EIP is created first
}

# Create route table for public subnets
resource "aws_route_table" "pub_rt" {
    vpc_id = aws_vpc.vpc.id
    route {
        cidr_block = "0.0.0.0/0"
        gateway_id = aws_internet_gateway.igw.id
    }
    tags = {
        Name = "${var.name}-pub_rt"
    }
}

# Create route table for private subnets
resource "aws_route_table" "pri_rt" {
    vpc_id = aws_vpc.vpc.id
    route {
        cidr_block = "0.0.0.0/0"
        gateway_id = aws_nat_gateway.ngw.id
    }
    tags = {
        Name = "${var.name}-pri_rt"
    }
}

# Creating route table association for public_subnet_1
resource "aws_route_table_association" "ass-public_subnet_1" {
    subnet_id      = aws_subnet.pub_sub1.id
    route_table_id = aws_route_table.pub_rt.id
}

# Creating route table association for public_subnet_2
resource "aws_route_table_association" "ass-public_subnet_2" {
    subnet_id      = aws_subnet.pub_sub2.id
    route_table_id = aws_route_table.pub_rt.id
}

# Creating route table association for private_subnet_1
resource "aws_route_table_association" "ass-private_subnet_1" {
    subnet_id      = aws_subnet.pri_sub1.id
    route_table_id = aws_route_table.pri_rt.id
}

# Creating route table association for private_subnet_2
resource "aws_route_table_association" "ass-private_subnet_2" {
    subnet_id      = aws_subnet.pri_sub2.id
    route_table_id = aws_route_table.pri_rt.id
}

#creating keypair RSA key of size 4096 bits

```

```

resource "tls_private_key" "key" {
  algorithm = "RSA"
  rsa_bits  = 4096
}

# Creating private key
resource "local_file" "private-key" {
  content      = tls_private_key.key.private_key_pem
  filename     = "${var.name}-key.pem"
  file_permission = 440
}

# Creating public key
resource "aws_key_pair" "public-key" {
  key_name    = "${var.name}-infra-key"
  public_key  = tls_private_key.key.public_key_openssh
}

```

Nexus Module

The nexus server serves as our online repository manager. Over here we would be creating two registries, first repository would be maven hosted repo where we would store the artefacts created by maven and the second registry would be Docker hosted where we would store the docker image of our java based application.

In this code, we created a security group resource block (resource "aws_security_group" "nexus-sg") for our nexus server which is opened on port 22 for only bastion host , other opened port include port 8081 for our maven hosted repository and port 8085 for docker hosted repository. next is to create the nexus server (resource "aws_instance" "nexus")on the public subnet together with the userdatascript that helps to install all dependencies the nexus server uses, also we added an elastic load balancers resource block (resource "aws_security_group" "nexus-elb-sg") and its security group(resource "aws_security_group" "nexus-elb-sg"). The nexus elb security group is opened on port 443 to interface with internet securely, while the load balancer listens on port 8081.Finally we created a record for our nexus server using the (resource "aws_route53_record" "nexus-record") block

```

# Creating Nexus Security group
resource "aws_security_group" "nexus-sg" {
  name          = "${var.name}-nexus-sg"
  description   = "Allow SSH, HTTP, HTTPS, nexus and docker "
  vpc_id        = var.vpc

  ingress {
    from_port    = 22
    to_port     = 22
    protocol    = "tcp"
    security_groups = [var.bastion-sg]
  }
  ingress {
    from_port    = 8081
    to_port     = 8081
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

```

ingress {
  from_port   = 8085
  to_port     = 8085
  protocol    = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}

egress {
  from_port   = 0
  to_port     = 0
  protocol    = "-1"
  cidr_blocks = ["0.0.0.0/0"]
}

tags = {
  Name = "${var.name}-nexus-sg"
}
}

# Data source to get the latest RedHat AMI
data "aws_ami" "redhat" {
  most_recent = true
  owners      = ["309956199498"] # RedHat's owner ID
  filter {
    name  = "name"
    values = ["RHEL-9.4.0_HVM-20240605-x86_64-82-Hourly2-GP3"]
  }
  filter {
    name  = "virtualization-type"
    values = ["hvm"]
  }
  filter {
    name  = "architecture"
    values = ["x86_64"]
  }
}

resource "aws_instance" "nexus" {
  ami                               = data.aws_ami.redhat.id
  # ami                             = "ami-07d4917b6f95f5c2a" # Red Hat
  Enterprise_Linux
  instance_type                     = "t2.medium"
  subnet_id                         = var.subnet-id
  vpc_security_group_ids           = [aws_security_group.nexus-sg.id]
  key_name                           = var.keypair
  associate_public_ip_address       = true
  user_data                          = local.userdata
  root_block_device {
    volume_size = 30    # Size in GB
    volume_type = "gp3" # General Purpose SSD (recommended)
    encrypted   = true  # Enable encryption (best practice)
  }

  tags = {
    Name = "${var.name}-nexus-server"
  }
}

```

```

}

# Create Security group for the nexus elb
resource "aws_security_group" "nexus-elb-sg" {
  name          = "${var.name}-nexus-elb-sg"
  description   = "Allow HTTPS"
  vpc_id        = var.vpc

  ingress {
    from_port    = 443
    to_port     = 443
    protocol    = "TCP"
    cidr_blocks = ["0.0.0.0/0"]
  }
  egress {
    from_port    = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = {
    Name = "${var.name}-nexus-elb-sg"
  }
}
# Create elastic Load Balancer for nexus
resource "aws_elb" "elb_nexus" {
  name          = "${var.name}-nexus-elb"
  subnets       = [var.subnet1_id, var.subnet2_id]
  security_groups = [aws_security_group.nexus-elb-sg.id]

  listener {
    instance_port      = 8081
    instance_protocol  = "HTTP"
    lb_port            = 443
    lb_protocol        = "HTTPS"
    ssl_certificate_id = var.acm_certificate_arn
  }
  health_check {
    healthy_threshold  = 3
    unhealthy_threshold = 5
    interval           = 30
    timeout             = 5
    target              = "TCP:8081"
  }
  instances          = [aws_instance.nexus.id]
  cross_zone_load_balancing = true
  idle_timeout       = 400
  connection_draining = true
  connection_draining_timeout = 400
  tags = {
    Name = "${var.name}-nexus-elb"
  }
}

# Create Route 53 record for bastion host
data "aws_route53_zone" "acp-zone" {
  name          = var.domain
  private_zone  = false
}

```

```

}

# Create Route 53 record for nexus server
resource "aws_route53_record" "nexus-record" {
  zone_id = data.aws_route53_zone.acp-zone.zone_id
  name    = "nexus.${var.domain}"
  type    = "A"
  alias {
    name          = aws_elb.elb_nexus.dns_name
    zone_id       = aws_elb.elb_nexus.zone_id
    evaluate_target_health = true
  }
}

resource "null_resource" "update_jenkins" {
  depends_on = [ aws_instance.nexus ]
  provisioner "local-exec" {
    command = <<EOF
#!/bin/bash
sudo cat <<EOT>> /etc/docker/daemon.json
{
  "insecure-registries" : ["${aws_instance.nexus.public_ip}:8085"]
}
EOT
EOF
    interpreter = ["bash", "-c"]
  }
}

```

Userdata script for the nexus server helps to install all dependencies for our nexus server

```

locals {
userdata = <<EOF
#!/bin/bash
sudo yum update -y
sudo yum install wget -y
sudo yum install java-1.8.0-openjdk.x86_64 -y
sudo mkdir /app && cd /app
sudo wget http://download.sonatype.com/nexus/3/nexus-3.23.0-03-unix.tar.gz
sudo tar -xvf nexus-3.23.0-03-unix.tar.gz
sudo mv nexus-3.23.0-03 nexus
sudo adduser nexus
sudo chown -R nexus:nexus /app/nexus
sudo chown -R nexus:nexus /app/sonatype-work
sudo cat <<EOT> /app/nexus/bin/nexus.rc
run_as_user="nexus"
EOT
sed -i '2s/-Xms2703m/-Xms512m/' /app/nexus/bin/nexus.vmoptions
sed -i '3s/-Xmx2703m/-Xmx512m/' /app/nexus/bin/nexus.vmoptions
sed -i '4s/-XX:MaxDirectMemorySize=2703m/-XX:MaxDirectMemorySize=512m/' /app/nexus/bin/nexus.vmoptions
sudo touch /etc/systemd/system/nexus.service
sudo cat <<EOT> /etc/systemd/system/nexus.service
[Unit]
Description=nexus service
After=network.target

```

```

[Service]
Type=forking
LimitNOFILE=65536
User=nexus
Group=nexus
ExecStart=/app/nexus/bin/nexus start
ExecStop=/app/nexus/bin/nexus stop
User=nexus
Restart=on-abort
[Install]
WantedBy=multi-user.target
EOT
sudo ln -s /app/nexus/bin/nexus /etc/init.d/nexus
sudo chkconfig --add nexus
sudo chkconfig --levels 345 nexus on
sudo service nexus start
#curl -Ls
https://download.newrelic.com/install/newrelic-cli/scripts/install.sh | bash
&& sudo NEW_RELIC_API_KEY=${var.nr-key} NEW_RELIC_ACCOUNT_ID=${var.nr-id}
/usr/local/bin/newrelic install -y
sudo hostnamectl set-hostname Nexus
EOF
}

```

Sonarqube server provision

Here we are using sonarqube to scan for code smell,misconfiguration on our application code.all dependencies for our sonarqube server are installed on the userdatascript

In this code, we created a security group resource block (resource "aws_security_group" "sonarqube_sg")for our sonarqube server which is opened on port 22 for only bastion host , other opened port include port 9000 which is sonarqube port opened for ingress to only sonarqube elastic load balancers.

Next is to create the Sonarqube server (resource "aws_instance" "sonarqube-server")on the public subnet together with the userdatascript that helps to install all dependencies the sonarqube server uses, also we added an elastic load balancers resource block (resource "aws_security_group" "elb-sonar-sg") and its security group(resource "aws_security_group" "elb-sonar-sg") . The sonarqube elb security group is opened on port 443 to interface with internet securely, while the load balancer listens on port 9000.Finally we created a record for our sonarqube server using the (rresource "aws_route53_record" "sonar-record") block

```

# Ubuntu AMI lookup
data "aws_ami" "ubuntu" {
  most_recent = true
  owners      = ["099720109477"] # Canonical
  filter {
    name  = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-*"]
  }
  filter {
    name  = "virtualization-type"
    values = ["hvm"]
  }
}

```

```

resource "aws_instance" "sonarqube-server" {
  ami                      = data.aws_ami.ubuntu.id # ubuntu in eu-west-2
  instance_type             = "t2.medium"
  key_name                  = var.key
  subnet_id                 = var.subnet_id
  associate_public_ip_address = true
  vpc_security_group_ids    = [aws_security_group.sonarqube_sg.id]
  root_block_device {
    volume_size = 20      # Size in GB
    volume_type = "gp3"   # General Purpose SSD (recommended)
    encrypted   = true    # Enable encryption (best practice)
  }
  user_data = local.userdata
  tags = {
    Name = "${var.name}-sonarqube-server"
  }
}

# Create sonarqube security group
resource "aws_security_group" "sonarqube_sg" {
  name          = "${var.name}-sonarqube-sg"
  description   = "Allow SSH and HTTPS"
  vpc_id        = var.vpc-id
  ingress {
    from_port    = 22
    to_port      = 22
    protocol     = "tcp"
    security_groups = [var.bastion]
  }
  ingress {
    from_port    = 9000
    to_port      = 9000
    protocol     = "tcp"
  }
  security_groups = [aws_security_group.elb-sonar-sg.id]
}
  egress {
    from_port    = 0
    to_port      = 0
    protocol     = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = {
    Name = "${var.name}-sonarqube-sg"
  }
}

#Create Security Group for Sonarqube Sever ELB
resource "aws_security_group" "elb-sonar-sg" {
  name          = "${var.name}-sonarqube-elb-sg"
  description   = "Allow HTTPS"
  vpc_id        = var.vpc-id

  ingress {
    from_port    = 443
    to_port      = 443
    protocol     = "tcp"
  }
}

```

```

        cidr_blocks = ["0.0.0.0/0"]
    }
    egress {
        from_port   = 0
        to_port     = 0
        protocol    = "-1"
        cidr_blocks = ["0.0.0.0/0"]
    }
    tags = {
        Name = "${var.name}-sonarqube-elb-sg"
    }
}

# Create Elastic load balancer for Sonarqube Server
resource "aws_elb" "elb-sonar" {
    name          = "${var.name}-elb-sonar"
    subnets       = var.public_subnets
    security_groups = [aws_security_group.elb-sonar-sg.id]

    listener {
        instance_port      = 9000
        instance_protocol  = "http"
        lb_port            = 443
        lb_protocol        = "https"
        ssl_certificate_id = var.acm_certificate_arn
    }

    health_check {
        healthy_threshold  = 2
        unhealthy_threshold = 2
        timeout           = 3
        target             = "TCP:9000"
        interval          = 30
    }

    instances          = [aws_instance.sonarqube-server.id]
    cross_zone_load_balancing = true
    idle_timeout       = 400
    connection_draining = true
    connection_draining_timeout = 400

    tags = {
        Name = "${var.name}-elb-sonar"
    }
}

# Create Route 53 Hosted Zone for Sonarqube
data "aws_route53_zone" "zone_id" {
    name      = var.domain
    private_zone = false
}

# Create Route 53 A Record for Sonarqube Server
resource "aws_route53_record" "sonar-record" {
    zone_id = data.aws_route53_zone.zone_id.zone_id
    name    = "sonar.${var.domain}"
    type    = "A"
}

```

```

alias {
    name          = aws_elb.elb-sonar.dns_name
    zone_id       = aws_elb.elb-sonar.zone_id
    evaluate_target_health = true
}
}

```

sonarqube userdatascript.

```

locals {
userdata = <<EOF
#!/bin/bash
sudo apt update -y
echo "****Firstly Modify OS Level values***"
sudo bash -c 'echo "
vm.max_map_count=262144
fs.file-max=65536
ulimit -n 65536
ulimit -u 4096" >> /etc/sysctl.conf'
sudo bash -c 'echo "
sonarqube - nofile 65536
sonarqube - nproc 4096" >> /etc/security/limits.conf'
echo "*****Install Java JDK*****"
sudo apt install openjdk-11-jdk -y
echo "*****Install PostgreSQL*****"
echo "*****The version of postgres currently is 14.5 which is not
supported so we have to download v12*****"
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo
apt-key add -
sudo apt-get update -y
sudo apt-get -y install postgresql-12 postgresql-contrib-12
echo "*****Enable and start, so it starts when system boots up*****"
sudo systemctl enable postgresql
sudo systemctl start postgresql
#Change default password of postgres user
sudo chpasswd <<<"postgres:Admin123@"
#Create user sonar without switching technically
sudo su -c 'createuser sonar' postgres
#Create SonarQube Database and change sonar password
sudo su -c "psql -c \"ALTER USER sonar WITH ENCRYPTED PASSWORD 'Admin123'\""
postgres
sudo su -c "psql -c \"CREATE DATABASE sonarqube OWNER sonar\"" postgres
sudo su -c "psql -c \"GRANT ALL PRIVILEGES ON DATABASE sonarqube to sonar\""
postgres
#Restart postgresql for changes to take effect
sudo systemctl restart postgresql
#Install SonarQube
sudo mkdir /sonarqube/
cd /sonarqube/
sudo wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-
8.6.0.39681.zip
sudo apt install unzip -y
sudo unzip sonarqube-8.6.0.39681.zip -d /opt/
sudo mv /opt/sonarqube-8.6.0.39681/ /opt/sonarqube

```

```

#Add group user sonarqube
sudo groupadd sonar
#Then, create a user and add the user into the group with directory
permission to the /opt/ directory
sudo useradd -c "SonarQube - User" -d /opt/sonarqube/ -g sonar sonar
#Change ownership of the directory to sonar
sudo chown sonar:sonar /opt/sonarqube/ -R
sudo bash -c 'echo "
sonar.jdbc.username=sonar
sonar.jdbc.password=Admin123
sonar.jdbc.url=jdbc:postgresql://localhost/sonarqube
sonar.search.javaOpts=-Xmx512m -Xms512m -XX:+HeapDumpOnOutOfMemoryError"
>> /opt/sonarqube/conf/sonar.properties'
#Configure such that SonarQube starts on boot up
sudo touch /etc/systemd/system/sonarqube.service
#Configuring so that we can run commands to start, stop and reload sonarqube
service
sudo bash -c 'echo "
[Unit]
Description=SonarQube service
After=syslog.target network.target
[Service]
Type=forking
ExecStart=/opt/sonarqube/bin/linux-x86-64/sonar.sh start
ExecStop=/opt/sonarqube/bin/linux-x86-64/sonar.sh stop
ExecReload=/opt/sonarqube/bin/linux-x86-64/sonar.sh restart
User=sonar
Group=sonar
Restart=always
LimitNOFILE=65536
LimitNPROC=4096
[Install]
WantedBy=multi-user.target" >> /etc/systemd/system/sonarqube.service'
#Enable and Start the Service
sudo systemctl daemon-reload
sudo systemctl enable sonarqube.service
sudo systemctl start sonarqube.service
#Install net-tools incase we want to debug later
sudo apt install net-tools -y
#Install nginx
sudo apt-get install nginx -y
#Configure nginx so we can access server from outside
sudo touch /etc/nginx/sites-enabled/sonarqube.conf
sudo bash -c 'echo "
server {
    listen 80;
    access_log  /var/log/nginx/sonar.access.log;
    error_log   /var/log/nginx/sonar.error.log;
    proxy_buffers 16 64k;
    proxy_buffer_size 128k;
    location / {
        proxy_pass  http://127.0.0.1:9000;
        proxy_next_upstream error timeout invalid_header http_500 http_502
http_503 http_504;
        proxy_redirect off;
        proxy_set_header Host          \$host;
        proxy_set_header X-Real-IP     \$remote_addr;
}
```

```

        proxy_set_header X-Forwarded-For \$proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto http;
    }
}" >> /etc/nginx/sites-enabled/sonarqube.conf'
#Remove the default configuration file
sudo rm /etc/nginx/sites-enabled/default
#Enable and restart nginx service
sudo systemctl enable nginx.service
sudo systemctl stop nginx.service
sudo systemctl start nginx.service
curl -Ls
https://download.newrelic.com/install/newrelic-cli/scripts/install.sh | bash
&& sudo NEW_RELIC_API_KEY=${var.nr-key} NEW_RELIC_ACCOUNT_ID=${var.nr-id}
/usr/local/bin/newrelic install -y
sudo hostnamectl set-hostname Sonarqube
sudo reboot
EOF
}

```

database server

Create a Multi az database. here data is persisted on the primary database and later synchronised to the secondary database. Also note that the security group of the database was open to access to only bastion host, prod and stage server on rds port 3306. this is because the stage and prod servers needs to have access to persist data also the bastion host needs have access for testing purposes.

```

# RDS Subnet Group
resource "aws_db_subnet_group" "db_subnet_group" {
  name      = "${var.name}-db_subnet"
  subnet_ids = [var.pri-sub-1, var.pri-sub-2] # Use private subnets for
security
  description = "Subnet group for Multi-AZ RDS deployment"

  tags = {
    Name = "${var.name}-db-Subnet-Group"
  }
}
data "vault_generic_secret" "vault-secret" {
  path = "secret/database"
}

resource "aws_db_instance" "mysql_database" {
  identifier          = "${var.name}-db"
  db_subnet_group_name = aws_db_subnet_group.db_subnet_group.name
  vpc_security_group_ids = [aws_security_group.RDS-sg.id]
  db_name              = "petadoption"
  # High Availability
  multi_az = true

  # Engine Settings
  engine           = "mysql"
  engine_version   = "5.7"
  instance_class    = "db.t3.micro"
}

```

```

parameter_group_name = "default.mysql5.7"

# Storage
allocated_storage = 20
storage_type      = "gp3"
storage_encrypted = true
# Credentials (Fetch from Vault Manager)
username = data.vault_generic_secret.vault-secret.data["username"]
password = data.vault_generic_secret.vault-secret.data["password"]
# Backup & Maintenance
skip_final_snapshot = true
# Security
publicly_accessible = false
deletion_protection = false
}

#RDS security group
resource "aws_security_group" "RDS-sg" {
  name          = "${var.name}-rds-sg"
  description   = "RDS Security group"
  vpc_id        = var.vpc-id

  ingress {
    description      = "mysqlport"
    from_port        = 3306
    to_port          = 3306
    protocol         = "tcp"
    security_groups = [var.bastion, var.stage-sg, var.prod-sg]
  }
  egress {
    from_port      = 0
    to_port        = 0
    protocol       = "-1"
    cidr_blocks   = ["0.0.0.0/0"]
  }
  tags = {
    Name = "${var.name}-rds-sg"
  }
}

```

Production server

This is is server that houses the contanarized application. here we used the auto-scaling group to deploy this environment this is also to increase availability so as to have a backup whenever we have a faulty production environment.

In the code, we have the security group of the production environment opened up on port 22 to allow access to only our bastion host and ansible server. this is because the ansible needs to have acces to configure the server whenever anew server is spinned up and the bastion host aswell needs access to access the server incase the administrator wants to check for any configuration on the server.

Also security group for the prod application load balancer was also created withaccess to only port 443 to allow https access through the load balancers to the docker host(production server) A record was also created for the production server.

```

#prod security group
resource "aws_security_group" "prod-sg" {
  name          = "${var.name}-prod-sg"
  description   = "prod Security group"
  vpc_id        = var.vpc-id
  ingress {
    description     = "SSH access from bastion"
    from_port      = 22
    to_port        = 22
    protocol       = "tcp"
    security_groups = [var.bastion, var.ansible]
  }
  ingress {
    description = "HTTP access from ALB"
    from_port   = 8080
    to_port     = 8080
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  egress {
    description = "Allow all outbound traffic"
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = {
    Name = "${var.name}-prod-sg"
  }
}
data "aws_ami" "redhat" {
  most_recent = true
  owners      = ["309956199498"] # RedHat's owner ID
  filter {
    name  = "name"
    values = ["RHEL-9*"]
  }
  filter {
    name  = "virtualization-type"
    values = ["hvm"]
  }
  filter {
    name  = "architecture"
    values = ["x86_64"]
  }
}
# Create Launch Template
resource "aws_launch_template" "prod_lnch_tmpl" {
  image_id      = data.aws_ami.redhat.id
  name_prefix   = "${var.name}-prod-web-tmpl"
  instance_type = "t2.medium"
  key_name      = var.key-name
  user_data     = base64encode(templatefile("./module/prod-env/docker-script.sh",
{
  nexus-ip           = var.nexus-ip,
  nr-key            = var.nr-key,

```

```

        nr-acct-id          = var.nr-acct-id
    }))}

network_interfaces {
    security_groups = [aws_security_group.prod-sg.id]
}
#user_data = ""
}

# Create Auto Scaling Group
resource "aws_autoscaling_group" "prod_autoscaling_grp" {
    name                  = "${var.name}-prod-asg"
    max_size              = 3
    min_size              = 1
    desired_capacity      = 1
    health_check_grace_period = 120
    health_check_type    = "EC2"
    force_delete          = true
    launch_template {
        id      = aws_launch_template.prod_lnch_tmpl.id
        version = "$Latest"
    }
    vpc_zone_identifier = [var.pri-subnet1, var.pri-subnet2]
    target_group_arns   = [aws_lb_target_group.prod-target-group.arn]

    tag {
        key          = "Name"
        value         = "${var.name}-prod-asg"
        propagate_at_launch = true
    }
}
# Created autoscaling group policy
resource "aws_autoscaling_policy" "prod-asg-policy" {
    name          = "asg-policy"
    adjustment_type = "ChangeInCapacity"
    autoscaling_group_name = aws_autoscaling_group.prod_autoscaling_grp.name
    policy_type   = "TargetTrackingScaling"
    target_tracking_configuration {
        predefined_metric_specification {
            predefined_metric_type = "ASGAverageCPUUtilization"
        }
        target_value = 70.0
    }
}

# Create Application Load Balancer for prod
resource "aws_lb" "prod_LB" {
    name          = "${var.name}-prod-LB"
    internal      = false
    load_balancer_type = "application"
    security_groups = [aws_security_group.prod-sg.id]
    subnets       = [var.pub-subnet1, var.pub-subnet2]
    tags = {
        Name = "${var.name}-prod-LB"
    }
}
#prod-elb security group

```

```

resource "aws_security_group" "prod-elb-sg" {
  name          = "${var.name}-prod-elb-sg"
  description   = "prod-elb Security group"
  vpc_id        = var.vpc_id
  ingress {
    description = "HTTP access from ALB"
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  egress {
    description = "Allow all outbound traffic"
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = {
    Name = "${var.name}-prod-elb-sg"
  }
}

#Create Target group for load Balancer
resource "aws_lb_target_group" "prod-target-group" {
  name          = "${var.name}-prod-tg"
  port          = 8080
  protocol     = "HTTP"
  vpc_id        = var.vpc_id
  target_type   = "instance"
  health_check {
    healthy_threshold  = 3
    unhealthy_threshold = 5
    interval           = 30
    timeout            = 5
    path               = "/"
  }
  tags = {
    Name = "${var.name}-prod-tg"
  }
}

# Create load balance listener for http
resource "aws_lb_listener" "prod_load_balancer_listener_http" {
  load_balancer_arn = aws_lb.prod_LB.arn
  port             = "80"
  protocol         = "HTTP"
  default_action {
    type      = "forward"
    target_group_arn = aws_lb_target_group.prod-target-group.arn
  }
}
# Create load balance listener for https
resource "aws_lb_listener" "prod_load_balancer_listener_https" {
  load_balancer_arn = aws_lb.prod_LB.arn
  port             = "443"
  protocol         = "HTTPS"
}

```

```

ssl_policy      = "ELBSecurityPolicy-2016-08"
certificate_arn = var.acm-cert-arn
default_action {
  type          = "forward"
  target_group_arn = aws_lb_target_group.prod-target-group.arn
}
}
# Create Route 53 record for prod server
data "aws_route53_zone" "team2-acp-zone" {
  name      = var.domain
  private_zone = false
}

# Create Route 53 record for prod server
resource "aws_route53_record" "prod-record" {
  zone_id = data.aws_route53_zone.team2-acp-zone.zone_id
  name    = "www.${var.domain}"
  type    = "A"
  alias {
    name          = aws_lb.prod_LB.dns_name
    zone_id       = aws_lb.prod_LB.zone_id
    evaluate_target_health = true
  }
}

```

prod server script

The major function for this production script is to install docker in the server and make it ready to receive docker image for our application deployment. It has in the script a file called script.sh which can only be executed when we want to pull the latest docker image from our nexus docker registry. the function of this script.sh is to login to the nexus docker registry,pull the latest image of our application stop and remove already running container, deploy the new image and the container on the server.

```

#!/bin/bash

#updatng the instance
sudo yum update -y
sudo yum upgrade -y

#install Docker and its dependencies, start Docker service
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo #downloads & adds
repo to instance
sudo yum install docker-ce -y

#add a registry to the Docker daemon configuration to allow
#insecure communication (without TLS verification) with a Docker registry on
port 8085
sudo cat <<EOT>> /etc/docker/daemon.json
{
  "insecure-registries" : ["${nexus-ip}:8085"]
}

```

EOT

```
#Starts the Docker service and enables it to run on boot.
#Add the ec2-user to the docker group, allowing them to run Docker commands.
sudo service docker start
sudo systemctl start docker
sudo systemctl enable docker

#add the ec2-user to the docker group, allowing them to run Docker commands.
sudo usermod -aG docker ec2-user

#create a shell script that manages a Docker container on the EC2 instance,
#pulling updates from a Nexus registry
#i.e. Configure docker to fetch image from Nexus
sudo mkdir /home/ec2-user/scripts
cat << EOF > "/home/ec2-user/scripts/script.sh"
#!/bin/bash

set -x

#Define Variables
IMAGE_NAME="${nexus-ip}:8085/petclinicapps"
CONTAINER_NAME="appContainer"
NEXUS_IP="${nexus-ip}:8085"

#Function to Login to dockerhub
authenticate_docker() {
    docker login --username=admin --password=admin123 \$NEXUS_IP
}

#Function to check for latest image on dockerhub
check_for_updates() {
    local latest_image=\$(docker pull \$IMAGE_NAME | grep "Status: Image is
up to date" | wc -l)
    if [ \$latest_image -eq 0 ]; then
        return 0
    else
        return 1
    fi
}

#Function to stop and remove the current container
#Function to deploy image in a container
update_container() {
    docker stop \$CONTAINER_NAME
    docker rm \$CONTAINER_NAME
    docker run -d --name \$CONTAINER_NAME -p 8080:8080 \$IMAGE_NAME
}

#Main Function
main() {
    authenticate_docker
    if check_for_updates; then
        update_container
        echo "Container upgraded to latest image."
    else
        echo "Up to date! No image update required. Exiting..."
    fi
}
```

```

        fi
    }
main
EOF

sudo chown -R ec2-user:ec2-user /home/ec2-user/scripts/script.sh
sudo chmod 777 /home/ec2-user/scripts/script.sh

#Restart Docker
sudo systemctl restart docker

#Install New Relic CLI, Set hostname for the instance
curl -Ls
https://download.newrelic.com/install/newrelic-cli/scripts/install.sh | bash
&& sudo NEW_RELIC_API_KEY="${nr-key}" NEW_RELIC_ACCOUNT_ID="${nr-acct-id}"
NEW_RELIC_REGION="EU" /usr/local/bin/newrelic install -y

#setting the host name
sudo hostnamectl set-hostname prod-instance
sudo reboot

```

Stage server

This is is server that houses the contanarized application. here we used the auto-scaling group to deploy this environment this is also to increase availability so as to have a backup whenever we have a faulty production environment.

In the code, we have the security group of the stage environment opened up on port 22 to allow access to only our bastion host and ansible server. this is because the ansible needs to have acces to configure the server whenever anew server is spinned up and the bastion host aswell needs access to access the server incase the administrator wants to check for any configuration on the server.

Also security group for the prod application load balancer was also created withaccess to only port 443 to allow https access through the load balancers to the docker host(stage server)
A record was also created for the production server.

```

#stage security group
resource "aws_security_group" "stage-sg" {
  name      = "${var.name}-stage-sg"
  description = "stage Security group"
  vpc_id    = var.vpc-id
  ingress {
    description      = "SSH access from bastion"
    from_port       = 22
    to_port         = 22
    protocol        = "tcp"
    security_groups = [var.bastion, var.ansible]
  }

  ingress {
    description      = "HTTP access from ALB"
    from_port       = 8080
    to_port         = 8080
    protocol        = "tcp"
  }
}

```

```

    cidr_blocks = ["0.0.0.0/0"]
    # security_groups = [aws_security_group.stage-elb-sg.id]
}
egress {
    description = "Allow all outbound traffic"
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
}
tags = {
    Name = "${var.name}-stage-sg"
}
}
data "aws_ami" "redhat" {
    most_recent = true
    owners      = ["309956199498"] # RedHat's owner ID
    filter {
        name  = "name"
        values = ["RHEL-9*"]
    }
    filter {
        name  = "virtualization-type"
        values = ["hvm"]
    }
    filter {
        name  = "architecture"
        values = ["x86_64"]
    }
}
# Create Launch Template
resource "aws_launch_template" "stage_lnch_tmpl" {
    image_id      = data.aws_ami.redhat.id
    name_prefix   = "${var.name}-stage-web-tmpl"
    instance_type = "t2.medium"
    key_name      = var.key-name
    user_data = base64encode(templatefile("./module/stage-env/docker-
script.sh", {
        nexus-ip          = var.nexus-ip,
        nr-key           = var.nr-key,
        nr-acct-id       = var.nr-acct-id
    }))
    network_interfaces {
        security_groups = [aws_security_group.stage-sg.id]
    }
    #user_data = ""
}

# Create Auto Scaling Group
resource "aws_autoscaling_group" "stage_autoscaling_grp" {
    name                  = "${var.name}-stage-asg"
    max_size              = 3
    min_size              = 1
    desired_capacity      = 1
    health_check_grace_period = 120
    health_check_type     = "EC2"
}

```

```

force_delete          = true
launch_template {
  id      = aws_launch_template.stage_lnch_tmpl.id
  version = "$Latest"
}
vpc_zone_identifier = [var.pri-subnet1, var.pri-subnet2]
target_group_arns   = [aws_lb_target_group.stage-target-group.arn]

tag {
  key        = "Name"
  value      = "${var.name}-stage-asg"
  propagate_at_launch = true
}
}

# Created autoscaling group policy
resource "aws_autoscaling_policy" "stage-asg-policy" {
  name          = "asg-policy"
  adjustment_type = "ChangeInCapacity"
  autoscaling_group_name = aws_autoscaling_group.stage_autoscaling_grp.name
  policy_type    = "TargetTrackingScaling"
  target_tracking_configuration {
    predefined_metric_specification {
      predefined_metric_type = "ASGAverageCPUUtilization"
    }
    target_value = 70.0
  }
}

# Create Application Load Balancer for stage
resource "aws_lb" "stage_LB" {
  name          = "${var.name}-stage-LB"
  internal      = false
  load_balancer_type = "application"
  security_groups = [aws_security_group.stage-elb-sg.id]
  subnets       = [var.pub-subnet1, var.pub-subnet2]
  tags = {
    Name = "${var.name}-stage-LB"
  }
}
#stage-elb security group
resource "aws_security_group" "stage-elb-sg" {
  name          = "${var.name}-stage-elb-sg"
  description = "stage-elb Security group"
  vpc_id       = var.vpc-id
  ingress {
    description = "HTTP access from ALB"
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
  egress {
    description = "Allow all outbound traffic"
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

```

```

    }
    tags = {
      Name = "${var.name}-stage-elb-sg"
    }
}

#Create Target group for load Balancer
resource "aws_lb_target_group" "stage-target-group" {
  name          = "${var.name}-stage-tg"
  port          = 8080
  protocol      = "HTTP"
  vpc_id        = var.vpc-id
  target_type   = "instance"
  health_check {
    healthy_threshold  = 3
    unhealthy_threshold = 5
    interval           = 30
    timeout            = 5
    path               = "/"
  }
  tags = {
    Name = "${var.name}-stage-tg"
  }
}

# Create load balance listener for http
resource "aws_lb_listener" "stage_load_balancer_listener_http" {
  load_balancer_arn = aws_lb.stage_LB.arn
  port             = "80"
  protocol         = "HTTP"
  default_action {
    type = "redirect"

    redirect {
      port      = 443
      protocol = "HTTPS"
      status_code = "HTTP_301"
    }
  }
}

# Create load balance listener for https
resource "aws_lb_listener" "stage_load_balancer_listener_https" {
  load_balancer_arn = aws_lb.stage_LB.arn
  port             = "443"
  protocol         = "HTTPS"
  ssl_policy       = "ELBSecurityPolicy-2016-08"
  certificate_arn = var.acm-cert-arn
  default_action {
    type      = "forward"
    target_group_arn = aws_lb_target_group.stage-target-group.arn
  }
}

# Create Route 53 record for stage server
data "aws_route53_zone" "team2-acp-zone" {
  name      = var.domain
  private_zone = false
}

```

```

# Create Route 53 record for stage server
resource "aws_route53_record" "stage-record" {
  zone_id = data.aws_route53_zone.team2-acp-zone.zone_id
  name    = "stage.${var.domain}"
  type    = "A"
  alias {
    name          = aws_lb.stage_LB.dns_name
    zone_id       = aws_lb.stage_LB.zone_id
    evaluate_target_health = true
  }
}

```

stage server script

The major function for this production script is to install docker in the server and make it ready to receive docker image for our application deployment. It has in the script a file called script.sh which can only be executed when we want to pull the latest docker image from our nexus docker registry. the function of this script.sh is to login to the nexus docker registry,pull the latest image of our application stop and remove already running container, deploy the new image and the container on the server.

```

#!/bin/bash

#updatng the instance
sudo yum update -y
sudo yum upgrade -y

#install Docker and its dependencies, start Docker service
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo #downloads & adds
repo to instance
sudo yum install docker-ce -y

#add a registry to the Docker daemon configuration to allow
#insecure communication (without TLS verification) with a Docker registry on
port 8085
sudo cat <<EOT>> /etc/docker/daemon.json
{
  "insecure-registries" : ["${nexus-ip}:8085"]
}
EOT

#Starts the Docker service and enables it to run on boot.
#Add the ec2-user to the docker group, allowing them to run Docker commands.
sudo service docker start
sudo systemctl start docker
sudo systemctl enable docker

#add the ec2-user to the docker group, allowing them to run Docker commands.
sudo usermod -aG docker ec2-user

```

```

#create a shell script that manages a Docker container on the EC2 instance,
pulling updates from a Nexus registry
#i.e. Configure docker to fetch image from Nexus
sudo mkdir /home/ec2-user/scripts
cat << EOF > "/home/ec2-user/scripts/script.sh"
#!/bin/bash

set -x

#Define Variables
IMAGE_NAME="${nexus-ip}:8085/petclinicapps"
CONTAINER_NAME="appContainer"
NEXUS_IP="${nexus-ip}:8085"

#Function to Login to dockerhub
authenticate_docker() {
    docker login --username=admin --password=admin123 \$NEXUS_IP
}

#Function to check for latest image on dockerhub
check_for_updates() {
    local latest_image=\$(docker pull \$IMAGE_NAME | grep "Status: Image is
up to date" | wc -l)
    if [ \$latest_image -eq 0 ]; then
        return 0
    else
        return 1
    fi
}

#Function to stop and remove the current container
#Function to deploy image in a container
update_container() {
    docker stop \$CONTAINER_NAME
    docker rm \$CONTAINER_NAME
    docker run -d --name \$CONTAINER_NAME -p 8080:8080 \$IMAGE_NAME
}

#Main Function
main() {
    authenticate_docker
    if check_for_updates; then
        update_container
        echo "Container upgraded to latest image."
    else
        echo "Up to date! No image update required. Exiting..."
    fi
}
main
EOF

sudo chown -R ec2-user:ec2-user /home/ec2-user/scripts/script.sh
sudo chmod 777 /home/ec2-user/scripts/script.sh

#Restart Docker
sudo systemctl restart docker

```

```

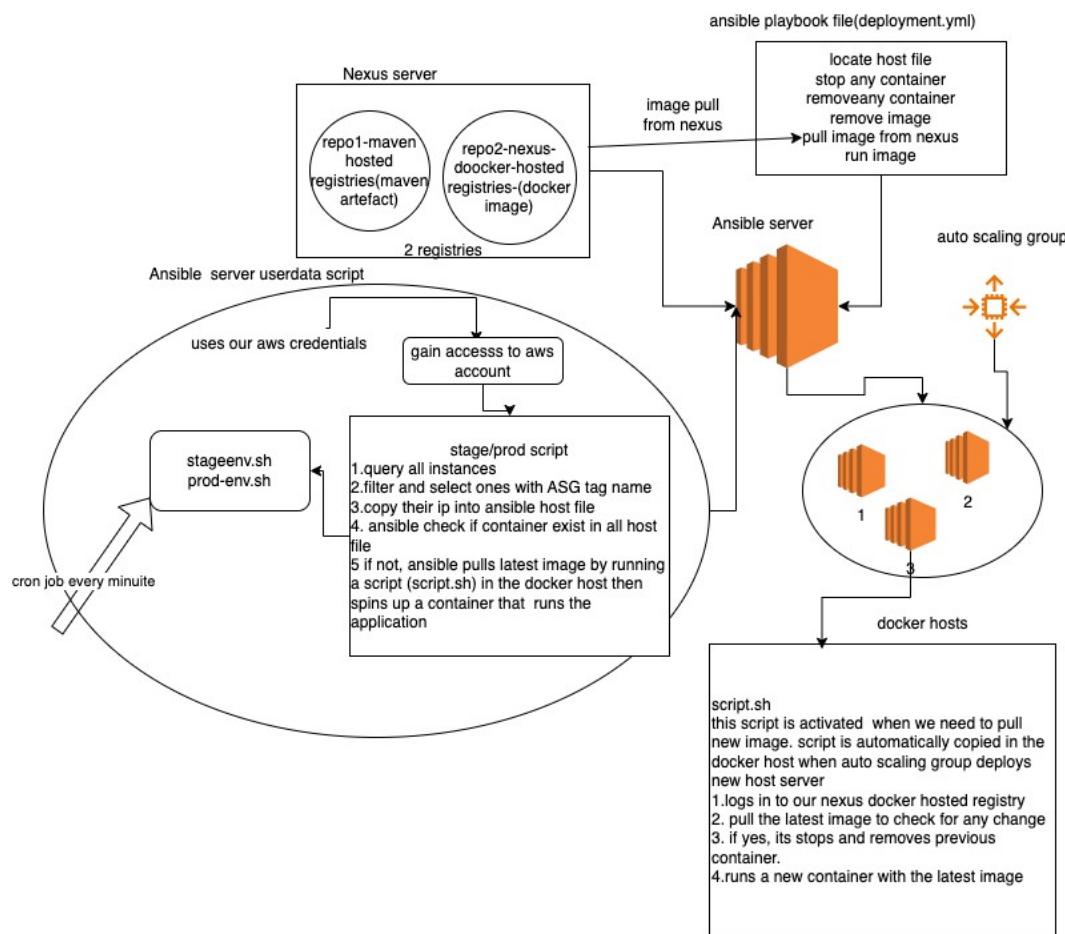
#Install New Relic CLI, Set hostname for the instance
curl -Ls
https://download.newrelic.com/install/newrelic-cli/scripts/install.sh | bash
&& sudo NEW_RELIC_API_KEY="${nr-key}" NEW_RELIC_ACCOUNT_ID="${nr-acct-id}"
NEW_RELIC_REGION="EU" /usr/local/bin/newrelic install -y

#setting the host name
sudo hostnamectl set-hostname stage-instance
sudo reboot

```

E. ANSIBLE CODE AND WORKFLOW EXPLANATION

Ansible server workflow



Here, the ansible server generally helps us to configure the docker host server and deploy the

application with the help of the ansible playbook.

Our Jenkinsfile houses various stages in our application pipeline, of which one of the stages is to deploy the application in our production/stage environment. When that stage is triggered, Our ansible server is triggered to run the ansible playbook. This is what happens when the ansible playbook is triggered:

It starts with the Ansible locating the host file where it would run the docker image of our application, It then stops and removes any running container together with the previous image if any.

Thereafter it finally pulls the latest image from the nexus docker hosted repository and runs the application.

NB : this process happens when ever there is a change in the code in our application pipeline.

Here comes the big question ???? how does Ansible update its host file with the latest ip addresses of the docker host (servers) based on the fact that all our docker host are deployed using autoscaling group that scales in when traffic reduces and scales out when traffic increases, how does Ansible learn about the new deployed servers on increased traffic?

This is how the ips of the docker servers are updated on the host file.

Ansible server is deployed along side an ansible userdata script. In the userdata script, we have set a cron job that runs every minute to execute our production/stage bashscript script every minute.

The production or stage script function is to query all the ec2 instances in our aws account and filter all ec2 instances created by our autoscaling group using the auto scaling group tag name with the help of our aws credentials already provided for it.

After a successful query is done, it copies these filtered ipaddress and replaces the host file of the ansible with the latest ip addresses it obtained after filtering.

In same script, ansible logins to each and every ip to check if there are any container running the application already, if yes, it skips that docker host to the next one and if it finds host with no latest container, it pulls the latest docker image using another script called (script.sh) that is stored already in the docker host when it was deployed by the auto scaling group.

The functon of this “script.sh“ is to login to our nexus docker registry to pull the latest version of the specified docker image in that repository, if newer version exist, downloads it, it stops and removes any old running container and redeploys the new version image in a new container. But if the output from the latest image version pulled by the docker has a status of”Image is up to date” that means that the old image is still the latest image and it does nothing and exits.

NB:Note that all these activity on our production/stage script runs every minute with the help of cron job that we set on our Ansible script.

so with this in place our Ansible host file is well updated every minute to get all host does some check to deploy our latest application image on our host servers if they do not exist.

Ansible [main.tf](#) file

ansible server was deployed using a redhat instance. We created a folder in our ansible module which we called “scripts” that houses our ansible playbook(deployment.yml), prduction and stage scripts. with the help of a null resource creation we were able to copy the above mentioned files to our s3 bucket created initially.

```
resource "null_resource" "copy_ansible_playbooks" {
  provisioner "local-exec" {
    command = <<EOT
      aws s3 cp --recursive ${path.module}/scripts/
s3://${var.s3Bucket}/ansible/
    EOT
  }
  depends_on = [time_sleep.wait_for_ansible]
}
```

Ansible security group was also created so as to allow ssh from the bastion host only, using (resource "aws_security_group" "ansible-sg")

IAM rols was also created and s3full access policy was attached to the role to enable ansible server have full access to perform operations on s3 bucket

Also IAM user was created to enable ansible perform operations /login to other ec2 servers to perform operation

```
# Data source to get the latest RedHat AMI
data "aws_ami" "redhat" {
  most_recent = true
  owners       = ["309956199498"] # RedHat's owner ID
  filter {
    name   = "name"
    values = ["RHEL-9*"]
  }
  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }
  filter {
    name   = "architecture"
    values = ["x86_64"]
  }
}
# create a time sleep resource to wait for the ansible server to be up and running
resource "time_sleep" "wait_for_ansible" {
  depends_on = [aws_instance.ansible-server]
  create_duration = "15s"
}
# null resource to copy the ansible playbooks folder into the s3 bucket
resource "null_resource" "copy_ansible_playbooks" {
  provisioner "local-exec" {
    command = <<EOT
      aws s3 cp --recursive ${path.module}/scripts/
s3://${var.s3Bucket}/ansible/
    EOT
  }
  depends_on = [time_sleep.wait_for_ansible]
```

```

}

# Create Ansible Server
resource "aws_instance" "ansible-server" {
  ami                  = data.aws_ami.redhat.id #rehat
  instance_type        = "t2.medium"
  iam_instance_profile= aws_iam_instance_profile.s3-bucket-instance-
profile.name
  vpc_security_group_ids = [aws_security_group.ansible-sg.id]
  key_name             = var.keypair
  subnet_id            = var.subnet_id
  user_data            = local.ansible_userdata
  root_block_device {
    volume_size = 20
    volume_type = "gp3"
    encrypted   = true
  }
  tags = {
    Name = "${var.name}-ansible-server"
  }
}

#Creating ansible security group
resource "aws_security_group" "ansible-sg" {
  name      = "${var.name}ansible-sg"
  description = "Allow ssh"
  vpc_id     = var.vpc

  ingress {
    description      = "sshport"
    from_port        = 22
    to_port          = 22
    protocol         = "tcp"
    security_groups = [var.bastion]
  }

  egress {
    from_port      = 0
    to_port        = 0
    protocol       = "-1"
    cidr_blocks   = ["0.0.0.0/0"]
  }
  tags = {
    Name = "${var.name}-ansible-sg"
  }
}

# IAM User
resource "aws_iam_user" "ansible-user" {
  name = "${var.name}-ansible-user"
}

resource "aws_iam_group" "ansible-group" {
  name = "${var.name}-ansible-group"
}

resource "aws_iam_access_key" "ansible-user-key" {
```

```

    user = aws_iam_user.ansible-user.name
}

resource "aws_iam_user_group_membership" "ansible-group-member" {
    user      = aws_iam_user.ansible-user.name
    groups   = [aws_iam_group.ansible-group.name]
}

resource "aws_iam_group_policy_attachment" "ansible-policy" {
    policy_arn = "arn:aws:iam::aws:policy/AmazonEC2FullAccess"
    group      = aws_iam_group.ansible-group.name
}

# Create IAM role for ansible server
resource "aws_iam_role" "s3-bucket-role" {
    name = "${var.name}-ansible-bucket-role"

    assume_role_policy = jsonencode({
        Version = "2012-10-17",
        Statement = [
            {
                Effect = "Allow",
                Principal = {
                    Service = "ec2.amazonaws.com"
                },
                Action = "sts:AssumeRole"
            }
        ]
    })
}

resource "aws_iam_role_policy_attachment" "ansible-bucket-role-attachment" {
    role      = aws_iam_role.s3-bucket-role.name
    policy_arn = "arn:aws:iam::aws:policy/AmazonS3FullAccess"
}

# Attach the role to the instance
resource "aws_iam_instance_profile" "s3-bucket-instance-profile" {
    name = "${var.name}-s3-bucket-instance-profile"
    role = aws_iam_role.s3-bucket-role.name
}

```

Production bashscript

This production environment script contains functions that queries all Ec2 instances in our AWS account to fish out all instances with the autoscaling group tag name as well as the instance ip addresses.

it fetches the ip, pushes the ip's into the Ansible inventory file, logs into them one by one to check if they are running the latest container image and if not, it executes a preinstalled script(script.sh) that pull the lates image from the nexus docker repository to run the application.

```

#!/bin/bash
set -x

```

```

#defineing our variables
AWSCLI_PATH='/usr/local/bin/aws'
INVENTORY_FILE='/etc/ansible/prod_hosts'
IPS_FILE='/etc/ansible/prod.lists'
ASG_NAME='team2-ad-prod-asg'
SSH_KEY_PATH='~/.ssh/id_rsa'
WAIT_TIME=20

#write our functions
#fetching the IPs
find_ips() {
    \$AWSCLI_PATH ec2 describe-instances \\
    --filters "Name>tag:aws:autoscaling:groupName,Values=\$ASG_NAME" \\
    --query
'Reservations[*].Instances[*].NetworkInterfaces[*].PrivateIpAddress' \\
    --output text > "\$IPS_FILE"
}
#update the inventory files
update_inventory() {
    echo "[webservers]" > "\$INVENTORY_FILE"
    while IFS= read -r instance; do
        ssh-keyscan -H "\$instance" >> ~/.ssh/known_hosts
        echo "\$instance ansible_user=ec2-user" >> "\$INVENTORY_FILE"
    done < "\$IPS_FILE"
    echo "Inventory updated succesfully"
}
#wait for some minutes
wait_for_seconds() {
    echo "Waiting for \$WAIT_TIME seconds..."
    sleep "\$WAIT_TIME"
}
# check if the docker container is running on all the instances in the ips
file by sshing into them one by one,
# if container is not running on the instance, then execute a script
#/home/ec2-user/scripts.sh to start the container
check_docker_container() {
    while IFS= read -r instance; do
        ssh -i "\$SSH_KEY_PATH" ec2-user@"\$instance" "docker ps | grep
appContainer" > /dev/null 2>&1
        if [ \$? -ne 0 ]; then
            echo "Container not running on \$instance. Starting container..."
            ssh -i "\$SSH_KEY_PATH" ec2-user@"\$instance" "bash /home/ec2-
user/scripts/script.sh"
        else
            echo "Container is running on \$instance."
        fi
    done < "\$IPS_FILE"
}
# Main function block
main() {
    find_ips
    update_inventory
    wait_for_seconds
    check_docker_container
}
# Execute main function
main

```

```
### End of script
```

Stage bash script

This Stage environment script contains functions that queries all Ec2 instances in our AWS account to fish out all instances with the autoscaling group tag name as well as the instance ip addresses.

it fetches the ip, pushes the ip's into the Ansible inventory file, logs into them one by one to check if they are running the latest container image and if not, it executes a preinstalled script(script.sh) that pull the latest image from the nexus docker repository to run the application.

```
#!/bin/bash
set -x

# Set up logging
LOG_FILE="/var/log/stage-bashscript.log"
exec > >(tee -a "$LOG_FILE") 2>&1
echo "Stage script started at $(date)"

# Defining our variables
AWSCLI_PATH='/usr/local/bin/aws'
INVENTORY_FILE='/etc/ansible/stage_hosts'
IPS_FILE='/etc/ansible/stage.lists'
ASG_NAME='auto-discov-stage-asg'
SSH_KEY_PATH='/home/ec2-user/.ssh/id_rsa' # Fixed: Using full path
WAIT_TIME=20

# Ensure directories exist
mkdir -p /etc/ansible
mkdir -p /home/ec2-user/.ssh
chown ec2-user:ec2-user /home/ec2-user/.ssh

# Write our functions

# Fetching the IPs
find_ips() {
    if ! $AWSCLI_PATH ec2 describe-instances \
        --filters "Name>tag:aws:autoscaling:groupName,Values=$ASG_NAME" \
        --query
    'Reservations[*].Instances[*].NetworkInterfaces[*].PrivateIpAddress' \
        --output text > "$IPS_FILE"; then
        echo "ERROR: Failed to fetch stage IPs" | tee -a "$LOG_FILE"
        return 1
    fi
    echo "Successfully fetched stage IPs"
}

# Update the inventory files
update_inventory() {
```

```

echo "[webservers]" > "$INVENTORY_FILE"
while IFS= read -r instance; do
    if [ -z "$instance" ]; then
        continue
    fi
    ssh-keyscan -H "$instance" >> /home/ec2-user/.ssh/known_hosts
2>/dev/null
    echo "$instance ansible_user=ec2-user
ansible_ssh_private_key_file=$SSH_KEY_PATH" >> "$INVENTORY_FILE"
    done < "$IPS_FILE"
    echo "Stage inventory updated successfully"
}

# Wait for some minutes
wait_for_seconds() {
    echo "Waiting for $WAIT_TIME seconds..."
    sleep "$WAIT_TIME"
}

# Check docker container status
check_docker_container() {
    while IFS= read -r instance; do
        if [ -z "$instance" ]; then
            continue
        fi

        if ! ssh -o StrictHostKeyChecking=no -i "$SSH_KEY_PATH" ec2-
user@"$instance" "docker ps | grep stageContainer" > /dev/null 2>&1; then
            echo "Stage container not running on $instance. Starting
container..." | tee -a "$LOG_FILE"
            if ! ssh -o StrictHostKeyChecking=no -i "$SSH_KEY_PATH" ec2-
user@"$instance" "bash /home/ec2-user/stage-scripts/script.sh"; then
                echo "ERROR: Failed to start stage container on $instance" |
tee -a "$LOG_FILE"
            fi
        else
            echo "Stage container is running on $instance." | tee -a
"$LOG_FILE"
        fi
        done < "$IPS_FILE"
}

# Main function block
main() {
    echo "Starting stage environment maintenance..."

    if ! find_ips; then
        exit 1
    fi

    if ! update_inventory; then
        exit 1
    fi

    wait_for_seconds

    if ! check_docker_container; then

```

```

        exit 1
    fi

    echo "Stage maintenance completed successfully at $(date)"
}

# Execute main function
main

### End of script

```

ansible playbook(deployment.yml file)

the ansible play book starts with identifying the hosts where the application would be deployed and variable file that contains information on docker registry.
this is followed by

```

---
- hosts: webservers
  become: true
  vars_files:
    - ansible_vars_file.yml
  tasks:
    - name: Stop any container running
      command: docker stop appContainer
      ignore_errors: yes
    - name: Remove stopped container
      command: docker rm appContainer
      ignore_errors: yes
    - name: Remove docker image
      command: docker rmi "{{ NEXUS_IP }}"/petclinicapps:latest
      ignore_errors: yes
    - name: Pull docker image from nexus and create a container
      shell: |
        sudo su -c "docker login --username=admin --password=admin123
{{ NEXUS_IP }}" ec2-user
        sudo su -c "docker pull "{{ NEXUS_IP }}"/petclinicapps:latest" ec2-
user
        sudo su -c "docker run -it -d --name appContainer -p 8080:8080
"{{ NEXUS_IP }}"/petclinicapps:latest" ec2-user

```

root [main.tf](#)

In this root [main.tf](#) file, we called out all the modules we already created so that terraform would utilise it to provision our infrastructure. here we have called out the bastion, vpc, sonarqube, nexus, ansible, database, production and stage environment modules

```

locals {
  name = "team2-ad"
}

module "bastion" {
  source  = "./module/bastion"
  name    = local.name
  vpc     = module.vpc.vpc_id
  keypair = module.vpc.public_key

```

```

privatekey = module.vpc.private_key
subnets = [module.vpc.pub_sub1_id, module.vpc.pub_sub2_id]
}
module "vpc" {
  source = "./module/vpc"
  name   = local.name
  az1    = "eu-west-2a"
  az2    = "eu-west-2b"
}
module "sonarqube" {
  source          = "./module/sonarqube"
  key            = module.vpc.public_key
  name           = local.name
  subnet_id      = module.vpc.pub_sub1_id
  bastion        = module.bastion.bastion-sg
  vpc_id         = module.vpc.vpc_id
  domain         = var.domain
  public_subnets = [module.vpc.pub_sub1_id, module.vpc.pub_sub2_id]
  acm_certificate_arn = data.aws_acm_certificate.team2-cert.arn
  nr-key         = var.nr-key
  nr-id          = var.nr-id
}
module "nexus" {
  source          = "./module/nexus"
  subnet_id      = module.vpc.pub_sub1_id
  keypair        = module.vpc.public_key
  name           = local.name
  vpc             = module.vpc.vpc_id
  baston_sg      = module.bastion.bastion-sg
  subnet1_id     = module.vpc.pub_sub1_id
  subnet2_id     = module.vpc.pri_sub2_id
  acm_certificate_arn = data.aws_acm_certificate.team2-cert.arn
  domain         = var.domain
  nr-key         = var.nr-key
  nr-id          = var.nr-id
}
module "ansible" {
  source      = "./module/ansible"
  name        = local.name
  keypair    = module.vpc.public_key
  subnet_id  = module.vpc.pri_sub1_id
  vpc         = module.vpc.vpc_id
  bastion    = module.bastion.bastion-sg
  private-key = module.vpc.private_key
  deployment  = "./module/ansible/deployment.yml" # Path to the deployment
file
  prod-bashscript = "./module/ansible/prod-bashscript.sh" # Path to the prod
bash script
  stage-bashscript = "./module/ansible/stage-bashscript.sh" # Path to the
stage bash script
  nexus-ip = module.nexus.nexus_ip
  nr-key = var.nr-key
  nr-acc-id = var.nr-id
}
module "database" {
  source      = "./module/database"
  name        = local.name

```

```

pri-sub-1 = module.vpc.pri_sub1_id
pri-sub-2 = module.vpc.pri_sub2_id
bastion   = module.bastion.bastion-sg
vpc-id    = module.vpc.vpc_id
stage-sg   = module.stage-env.stage-sg
prod-sg    = module.prod-env.prod-sg
}
module "prod-env" {
  source      = "./module/prod-env"
  name        = local.name
  vpc-id      = module.vpc.vpc_id
  bastion     = module.bastion.bastion-sg
  key-name    = module.vpc.public_key
  pri-subnet1 = module.vpc.pri_sub1_id
  pri-subnet2 = module.vpc.pri_sub2_id
  pub-subnet1 = module.vpc.pub_sub1_id
  pub-subnet2 = module.vpc.pub_sub2_id
  acm-cert-arn = data.aws_acm_certificate.team2-cert.arn
  domain      = var.domain
  nexus-ip    = module.nexus.nexus_ip
  nr-key      = var.nr-key
  nr-acct-id  = var.nr-id
  ansible     = module.ansible.ansible_sg
}
module "stage-env" {
  source      = "./module/stage-env"
  name        = local.name
  vpc-id      = module.vpc.vpc_id
  bastion     = module.bastion.bastion-sg
  key-name    = module.vpc.public_key
  pri-subnet1 = module.vpc.pri_sub1_id
  pri-subnet2 = module.vpc.pri_sub2_id
  pub-subnet1 = module.vpc.pub_sub1_id
  pub-subnet2 = module.vpc.pub_sub2_id
  acm-cert-arn = data.aws_acm_certificate.team2-cert.arn
  domain      = var.domain
  nexus-ip    = module.nexus.nexus_ip
  nr-key      = var.nr-key
  nr-acct-id  = var.nr-id
  ansible     = module.ansible.ansible_sg
}
data "aws_acm_certificate" "team2-cert" {
  domain    = var.domain
  statuses  = ["ISSUED"]
}

```

root [provider.tf](#)

This is the document that tells terraform what cloud provider we are using, the profile as well as the region of deployment.

it also provides information on the provider for our vault server as well as the token to be used to log into the vault server.

This is also where we configured our terraform to use the created s3 bucket as a backend to store our terraform state file.

```
provider "aws" {
```

```

region  = "eu-west-2"
# profile = "pet-adoption"

}

provider "vault" {
  address = "https://vault.bolatitoadegegoroye.top"
  token   = "s.654dBIbmq4zPtdgeyysk3VpM"
}

terraform {
  backend "s3" {
    bucket      = "team2-bucket-pet-adoption"
    key         = "infrastructure/terraform.tfstate"
    region      = "eu-west-2"
    use_lockfile = true
    encrypt     = true
  }
}

```

F. INFRASTRUCTURE JENKINSFILE EXPLANATION

the jenkins pipelines starts with setting up agent where the pipeline would run, so here we set up any agent which refers to any available agent which is our jenkins server.

next we setup tools that would be used by the pipeline. her we setup terraform as a tool that would be used to provision the infrastructure.

Next we set up trigger which we set up to run every midnight. also we set up environment variable like the SLACKCHANNEL where the team would receive notification about the success or failure of any build, and SLACKCREDENTIALS , wherewe set the crdentials that would be used for slack notification

After that, here comes the stages of the infrastructure pipeline.

It starts with infrastructure scan with checkov , then followed by terraform commnd to provision infrastructure.afterwhich the notification on slack for successful or unsuccessful build on slack.

```

pipeline {
  agent any
  tools {
    terraform 'terraform'
  }
  parameters {
    choice(name: 'action', choices: ['apply', 'destroy'], description:
'Select the action to perform')
  }
  triggers {
    cron('H 0 * * *') // Runs every day at midnight
  }
  environment {
    SLACKCHANNEL = 'D08B6QQ2T50' //MY CHANNEL ID
    SLACKCREDENTIALS = credentials('slack')
  }
  stages {
    stage('IAC Scan') {

```

```

        steps {
            script {
                sh 'pip install pipenv'
                sh 'pipenv run pip install checkov'
                def checkovStatus = sh(script: 'pipenv run checkov -d . -o cli --output-file checkov-results.txt --quiet', returnStatus: true)
                junit allowEmptyResults: true, testResults: 'checkov-results.txt'
                    // if (checkovStatus != 0) {
                    //     error 'Checkov found some issues'
                    // }
                }
            }
        }
    stage('Terraform Init') { // Fixed spelling
        steps {
            sh 'terraform init'
        }
    }
    stage('Terraform format') {
        steps {
            sh 'terraform fmt --recursive'
        }
    }
    stage('Terraform validate') {
        steps {
            sh 'terraform validate'
        }
    }
    stage('Terraform plan') {
        steps {
            sh 'terraform plan'
        }
    }
    stage('Terraform action') {
        steps {
            script {
                sh "terraform ${action} -auto-approve"
            }
        }
    }
}
post {
    always {
        script {
            slackSend(
                channel: SLACKCHANNEL,
                color: currentBuild.result == 'SUCCESS' ? 'good' :
                'danger',
                message: "Job '${env.JOB_NAME} [${env.BUILD_NUMBER}]' (${env.BUILD_URL}) has been completed."
            )
        }
    }
    failure {
        slackSend(
            channel: SLACKCHANNEL,

```

```

        color: 'danger',
        message: "Job '${env.JOB_NAME} [${env.BUILD_NUMBER}]' has
failed. Check console output at ${env.BUILD_URL}."
    )
}
success {
    slackSend(
        channel: SLACKCHANNEL,
        color: 'good',
        message: "Job '${env.JOB_NAME} [${env.BUILD_NUMBER}]'
completed successfully. Check console output at ${env.BUILD_URL}."
    )
}
}
}

```

G. INFRASTRUCTURE PROVISIONING SETUP FOR S3,VAULT AND JENKINS

After writing the code for infrastructure deployment, terraform was used to provision the infrastructure. before this provisioning, we set up our domain to be up and running.

First we purchase and registered a domain (edenboutique.space) at www.domain.com

The screenshot shows the 'Domains' section of the Domain.com website. The left sidebar has a navigation menu with options: Home, Websites, Email & Office, Domains (which is highlighted in teal), Add a Domain, Start Transfer In, Custom Nameservers, and Reports. The main content area is titled 'Domains' and features a search bar labeled 'Search domain names'. Below the search bar, there is a button 'ADVANCED VIEW'. A list of domains is displayed, with 'edenboutique.space' being the first item. The domain status is shown as 'Active' with a green checkmark. Below the domain name, it says 'Renewal Date : 24/05/2026' and 'Auto Renew Enabled'. To the right of the domain name is a 'SETTINGS' button. At the top of the page, there is a header with the 'DOMAIN.COM' logo, user information (Franklin), and a 'FO' button.

Create a route 53 hosted zone

us-east-1.console.aws.amazon.com/route53/v2/hostedzones?region=us-east-1#ListRecordSets/Z0092163ETAORMR2P7EQ

Route 53 > Hosted zones > edenboutique.space

Hosted zone details

Records (2) DNSSEC signing Hosted zone tags (0)

Records (2) Info

Record name	Type	Routing policy	Alias	Value/Route traffic to	TTL (s)	Health
edenboutique.space	NS	Simple	-	No ns-1304.awsdns-35.org. ns-869.awsdns-44.net. ns-2042.awsdns-63.co.uk. ns-422.awsdns-52.com.	172800	-
edenboutique.space	SOA	Simple	-	No ns-1304.awsdns-35.org. aws...	900	-

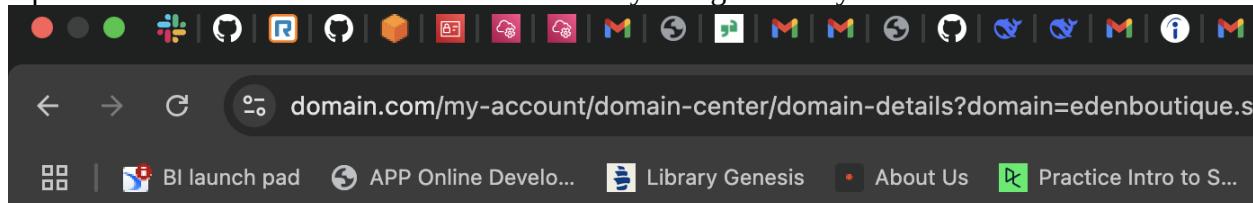
0 records selected

Select a record to

Route 53

- Dashboard
- Hosted zones
- Health checks
- Profiles New
- IP-based routing
- Traffic flow
- Domains
- Resolver

Update the hosted zone name server with where you registered your domain



- [Home](#)
- [Websites](#)
- [Email & Office](#)
- [Domains](#)
- [Add a Domain](#)
- [Start Transfer In](#)
- [Custom Nameservers](#)
- [Reports](#)
- [Security](#)

[RENEWAL CENTER](#)

Transfer or Move Domain

Move this domain to another account, another person,

Advanced Tools

Advanced tools offer more control and allow you to customize original nameserver settings to benefit from the suite of Domain.com services.

Nameservers (DNS)	MANAGE
NS-1304.AWSDNS-35.ORG	
NS-869.AWSDNS-44.NET	
NS-2042.AWSDNS-63.CO.UK	
NS-422.AWSDNS-52.COM	

Looking for custom nameservers?
Create them on the [Custom Nameservers](#) section.

[Legal](#) | [Privacy Policy](#) | [Terms of Use](#) | [Help](#)

Next, we provision the s3 bucket as well as the jenkins and vault server. the command to

provision this are all embedded in the s3 create-remote-state script.

On your terminal, run the “create-remote-state.sh” to provision the vault and Jenkins server

```
26  # # Enable versioning
27  echo "📦 Enabling versioning for S3 bucket..."
28  if ! aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-configuration Status=Enabled --re
29  | handle_error "Failed to enable versioning for S3 bucket '$BUCKET_NAME'." 
30 fi
31 echo "✅ Versioning enabled successfully."
32
33 echo "🚀 S3 Remote State Management Setup Complete!"
34 echo "🌐 S3 Bucket: $BUCKET_NAME"
35
36 # provision the vault and jenkins server
37 echo "🔧 Provisioning Vault and Jenkins server..."  
1
38 cd ./vault-jenkins
39 terraform init
40 terraform fmt --recursive
41 terraform validate
42 terraform apply --auto-approve
43 terraform output
44 echo "✅ Vault and Jenkins server provisioned successfully."
45
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
1487 cd autodiscovery-proj
1488 cd vault-jenkins
1489 ssh -i auto-discov-key.pem ec2-user@18.170.218.148\n
1490 ssh -i auto-discov-key.pem ec2-user@18.170.218.148\n
1491 brew tap hashicorp/tap\ncore install hashicorp/tap/terraform
1492 terraform --version
franklinonyia@Franklins-MacBook-Pro auto-discovery-mayist %
* History restored  
2
```

The operation couldn't be completed. Unable to locate a Java Runtime.
Please visit <http://www.java.com> for information on installing Java.

```
franklinonyia@Franklins-MacBook-Pro auto-discovery-mayist % cd autodiscovery-proj
franklinonyia@Franklins-MacBook-Pro autodiscovery-proj % sh create-remote-state.sh
```

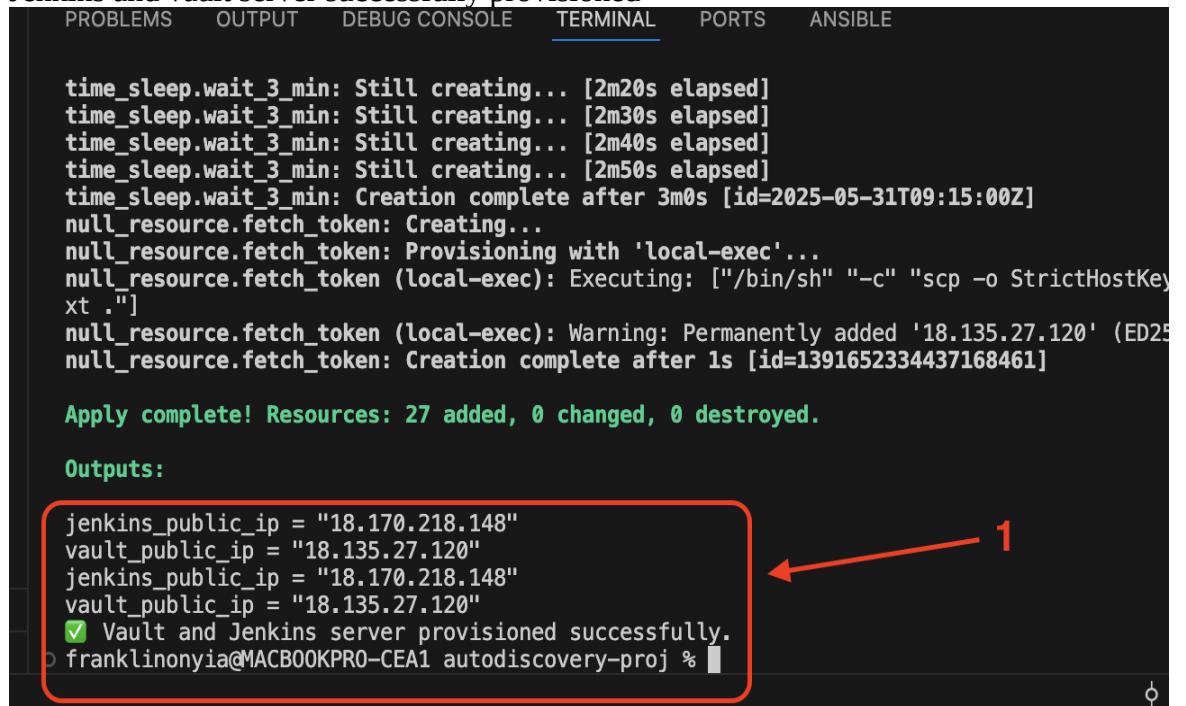
Bucket, vault and Jenkins provision ongoing

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    ANSIBLE
```

```
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/onyiafranklin/autodiscovery-proj.git
  2a48643..689f8cb feature/ebuka -> feature/ebuka
franklinonyia@MACBOOKPRO-CEA1 autodiscovery-proj % sh create-remote-state.sh
🔧 Creating S3 bucket: bucket-pet-adoption...
http://bucket-pet-adoption.s3.amazonaws.com/
✅ S3 bucket 'bucket-pet-adoption' created successfully.
📦 Enabling versioning for S3 bucket...
✅ Versioning enabled successfully.
🚀 S3 Remote State Management Setup Complete!
🌐 S3 Bucket: bucket-pet-adoption
🔧 Provisioning Vault and Jenkins server...
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/local from the dependency lock file
- Reusing previous version of hashicorp/null from the dependency lock file
- Reusing previous version of hashicorp/tls from the dependency lock file
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of hashicorp/time from the dependency lock file
- Using previously-installed hashicorp/null v3.2.4
- Using previously-installed hashicorp/tls v4.1.0
- Using previously-installed hashicorp/aws v5.98.0
```

φ onyiafranklin (1 week)

- Jenkins and vault server successfully provisioned



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ANSIBLE

time_sleep.wait_3_min: Still creating... [2m20s elapsed]
time_sleep.wait_3_min: Still creating... [2m30s elapsed]
time_sleep.wait_3_min: Still creating... [2m40s elapsed]
time_sleep.wait_3_min: Still creating... [2m50s elapsed]
time_sleep.wait_3_min: Creation complete after 3m0s [id=2025-05-31T09:15:00Z]
null_resource.fetch_token: Creating...
null_resource.fetch_token: Provisioning with 'local-exec'...
null_resource.fetch_token (local-exec): Executing: ["#!/bin/sh" "-c" "scp -o StrictHostKeyChecking=no ./jenkins-key.ppk root@18.135.27.120:/root/.ssh/authorized_keys"]
null_resource.fetch_token (local-exec): Warning: Permanently added '18.135.27.120' (ED25519) to the list of known hosts.
null_resource.fetch_token: Creation complete after 1s [id=1391652334437168461]

Apply complete! Resources: 27 added, 0 changed, 0 destroyed.

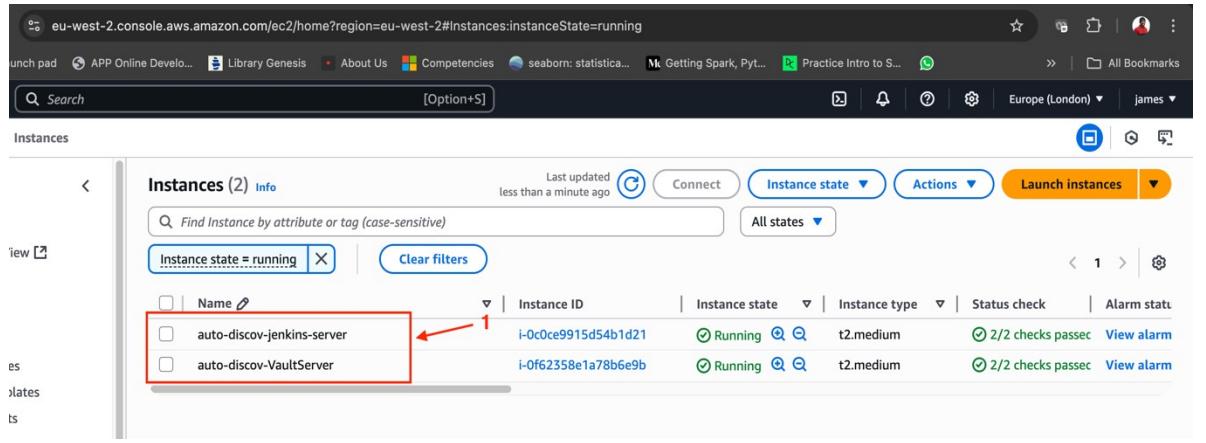
Outputs:

jenkins_public_ip = "18.170.218.148"
vault_public_ip = "18.135.27.120"
jenkins_public_ip = "18.170.218.148"
vault_public_ip = "18.135.27.120"
Vault and Jenkins server provisioned successfully.

franklinonyia@MACBOOKPRO-CEA1 autodiscovery-proj %

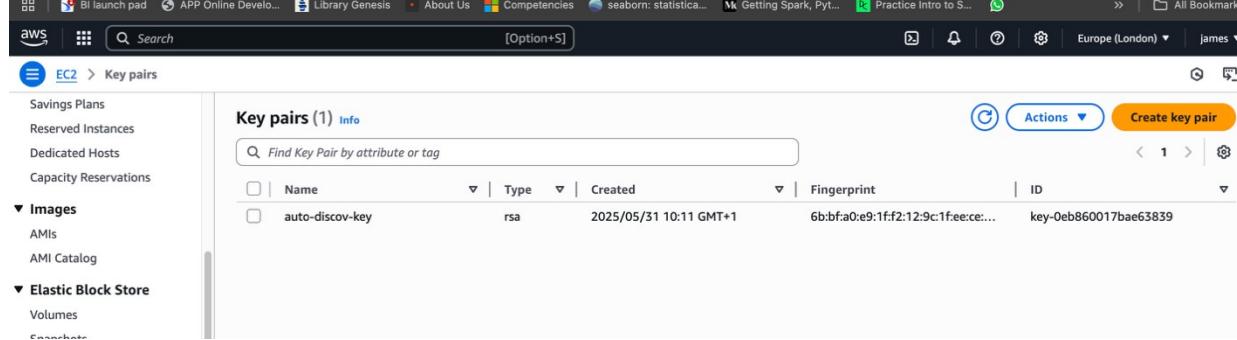
```

view Jenkins and vault server on the console



Instances (2) <small>Info</small>					
		Last updated less than a minute ago	<input type="button" value="Connect"/>	<input type="button" value="Instance state"/>	<input type="button" value="Actions"/>
<input type="text"/> Find Instance by attribute or tag (case-sensitive)		<input type="button" value="Clear filters"/>			
<input type="checkbox"/> Instance state = running <input type="button" value="X"/>		<input type="button" value="All states"/>			
Name	Instance ID	Instance state	Instance type	Status check	Alarm stats
<input type="checkbox"/> auto-discov-jenkins-server	i-0cce9915d54b1d21	<input checked="" type="radio"/> Running <input type="button" value="View details"/>	t2.medium	<input checked="" type="radio"/> 2/2 checks passed <input type="button" value="View alarm"/>	<input type="button" value="View alarm"/>
<input type="checkbox"/> auto-discov-VaultServer	i-0f62358e1a78b6e9b	<input checked="" type="radio"/> Running <input type="button" value="View details"/>	t2.medium	<input checked="" type="radio"/> 2/2 checks passed <input type="button" value="View alarm"/>	<input type="button" value="View alarm"/>

See elastic load balancers for the jenkins and vault .



Key pairs (1) <small>Info</small>					
<input type="text"/> Find Key Pair by attribute or tag					
Name	Type	Created	Fingerprint	ID	<input type="button" value="Actions"/>
<input type="checkbox"/> auto-discov-key	rsa	2025/05/31 10:11 GMT+1	6bbfa0:e9:1ff2:12:9c:1f:ee:ce:...	key-0eb860017bae63839	<input type="button" value="Create key pair"/>

Load balancers (1/2)						
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.						
<input type="button" value="Actions"/> <input type="button" value="Create load balancer"/>						
<input type="text" value="Filter load balancers"/>						
Name	DNS name	State	VPC ID	Availability Zones	Type	
elb-jenkins	elb-jenkins-1163844456.eu...	-	vpc-0dc620d69ea05bcc9	2 Availability Zones	classic	
vault-elb	vault-elb-1217819938.eu...	-	vpc-0dc620d69ea05bcc9	2 Availability Zones	classic	

view issued domain certificate

Certificates (1)						
<input type="button" value="Delete"/> <input type="button" value="Manage expiry events"/> <input type="button" value="Import"/> <input type="button" value="Request"/>						
Certificate ID	Domain name	Type	Status			
576c170f-a837-45c5-a7a9-361653f332de	edenboutique.space	Amazon Issued	Issued			

see certificate status for domain and sub domain on console.

Certificate status				
Identifier	576c170f-a837-45c5-a7a9-361653f332de			
ARN	arn:aws:acm:eu-west-2:896035377516:certificate/576c170f-a837-45c5-a7a9-361653f332de			
Type	Amazon Issued			
Status	Issued			
Domains (2)				
<input type="button" value="Create records in Route 53"/> <input type="button" value="Export to CSV"/>				
Domain	Status	Renewal status	Type	CNAME name
edenboutique.space	Success	-	CNAME	_4fd7843ac4259ba1a829d8477af6ce.
*.edenboutique.space	Success	-	CNAME	_4fd7843ac4259ba1a829d8477af6ce.

Details
view records created for vault and Jenkins

Route 53

Hosted zones

Records (5)

Record name	Type	Value/Route traffic to	TTL	Health
edenboutique.space	NS	ns-1304.awsdns-35.org. ns-869.awsdns-44.net. ns-2042.awsdns-63.co.uk. ns-422.awsdns-52.com.	172800	-
edenboutique.space	SOA	No	900	-
_4fd7843ac4259ba1a829d8477af65c06.edenboutique.space	CNAME	No	_08d7e07a6b7615b1fcece6...	60
jenkins.edenboutique.space	A	Yes	elb-jenkins-1163944456.eu...	-
vault.edenboutique.space	A	Yes	vault-elb-1217819938.eu-w...	-

View the created bucket

Amazon S3

General purpose buckets

Name	AWS Region	IAM Access Analyzer	Creation date
bucket-pet-adoption	Europe (London) eu-west-2	View analyzer for eu-west-2	May 31, 2025, 10:11:23 (UTC+01:00)

view the directory in the bucket.

Ansible file path :this houses all our ansible deployment file, stage and prod script copied

Infrastructure file path:houses the terraform state file

The screenshot shows the AWS S3 console interface. On the left, a sidebar titled 'Amazon S3' lists 'General purpose buckets' (Directory buckets, Table buckets, Access Grants, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, IAM Access Analyzer for S3), 'Storage Lens' (Dashboards, Storage Lens groups, AWS Organizations settings), and a 'Feature spotlight' section. The main area displays the 'bucket-pet-adoption' bucket. The 'Info' tab is selected. The 'Objects' tab is active, showing three objects: 'ansible/' (Folder), 'infrastructure/' (Folder), and 'vault-jenkins/' (Folder). Each object has a 'Copy S3 URI' button. A search bar at the top of the object list allows filtering by prefix. Below the search bar are buttons for 'Actions' (with dropdown options like 'Create folder' and 'Upload'), 'Show versions' (disabled), and navigation controls (back, forward, search, refresh).

content of the ansible folder

This screenshot shows the contents of the 'ansible/' folder within the 'bucket-pet-adoption' bucket. The sidebar on the left is identical to the previous screenshot. The main area shows the 'ansible/' folder with three objects: 'deployment.yml', 'prod-bashscript.sh', and 'stage-bashscript.sh'. Each object has a 'Copy S3 URI' button. A search bar at the top of the object list allows filtering by prefix. Below the search bar are buttons for 'Actions' (with dropdown options like 'Create folder' and 'Upload'), 'Show versions' (disabled), and navigation controls (back, forward, search, refresh).

Name	Type	Last modified	Size	Storage class
deployment.yml	yml	May 31, 2025, 11:19:36 (UTC+01:00)	777.0 B	Standard
prod-bashscript.sh	sh	May 31, 2025, 11:19:36 (UTC+01:00)	2.6 KB	Standard
stage-bashscript.sh	sh	May 31, 2025, 11:19:36 (UTC+01:00)	2.7 KB	Standard

vault sign in

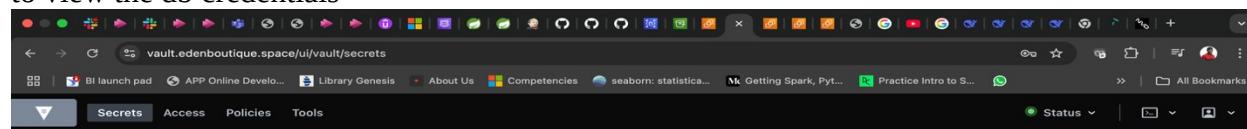


Vault secret create token that would be used to sign in to vault

```
$ docker-script.sh .../stage-env
$ docker-script.sh .../prod-env
```

```
autodiscovery-proj > vault-jenkins > token.txt
1 $arrE5QniIMMk2EMPGpnpkKhWx
2 +
```

view the secret folder where vault stored our database credentials. Open the secret to view the db credentials



Secrets Engines

		Enable new engine +
↳	cubbyhole/	...
↳	secret	...

kv_0650cbc0

The screenshot shows a web browser window with the URL `vault.edenboutique.space/ui/vault/secrets/secret/list`. The page title is "secret". There are tabs for "Secrets" and "Configuration". A search bar contains the placeholder "Filter secrets". A single secret named "database" is listed. On the right side, there are buttons for "Create secret" and three dots for more options.

view secrets

The screenshot shows a web browser window with the URL `vault.edenboutique.space/ui/vault/secrets/secret/show/database`. The page title is "database". There is a "JSON" button. A table lists two key-value pairs:

Key	Value
password	petclinic
username	petclinic

On the right side, there are buttons for "Delete secret", "Copy secret", and "Edit secret".

Next is to open the Jenkins using the url “Jenkins.edenboutique.space”
the login to Jenkins server to get the admin password on the
/var/lib/Jenkins/secret/initialAdminPassword file path

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

Continue

To get the password, ssh into the Jenkins server

```
● franklinonyia@MACBOOKPRO-CEA1 autodiscovery-proj % cd vault-jenkins
○ franklinonyia@MACBOOKPRO-CEA1 vault-jenkins % ssh -i auto-discov-key.pem ec2-user@18.170.218.148
The authenticity of host '18.170.218.148 (18.170.218.148)' can't be established.
ED25519 key fingerprint is SHA256:tkCV3chxIzJ+lr4gI7MtLz0amf6goqViLQ21u3QoA8k.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '18.170.218.148' (ED25519) to the list of known hosts.
Register this system with Red Hat Insights: rhc connect
```

Example:
rhc connect --activation-key <key> --organization <org>

The rhc client and Red Hat Insights will enable analytics and additional management capabilities on your system.
View your connected systems at <https://console.redhat.com/insights>

You can learn more about how to register your system
using rhc at <https://red.ht/registration>

[ec2-user@jenkins ~]\$

1

2

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

1

Please copy the password from either location and paste it below.

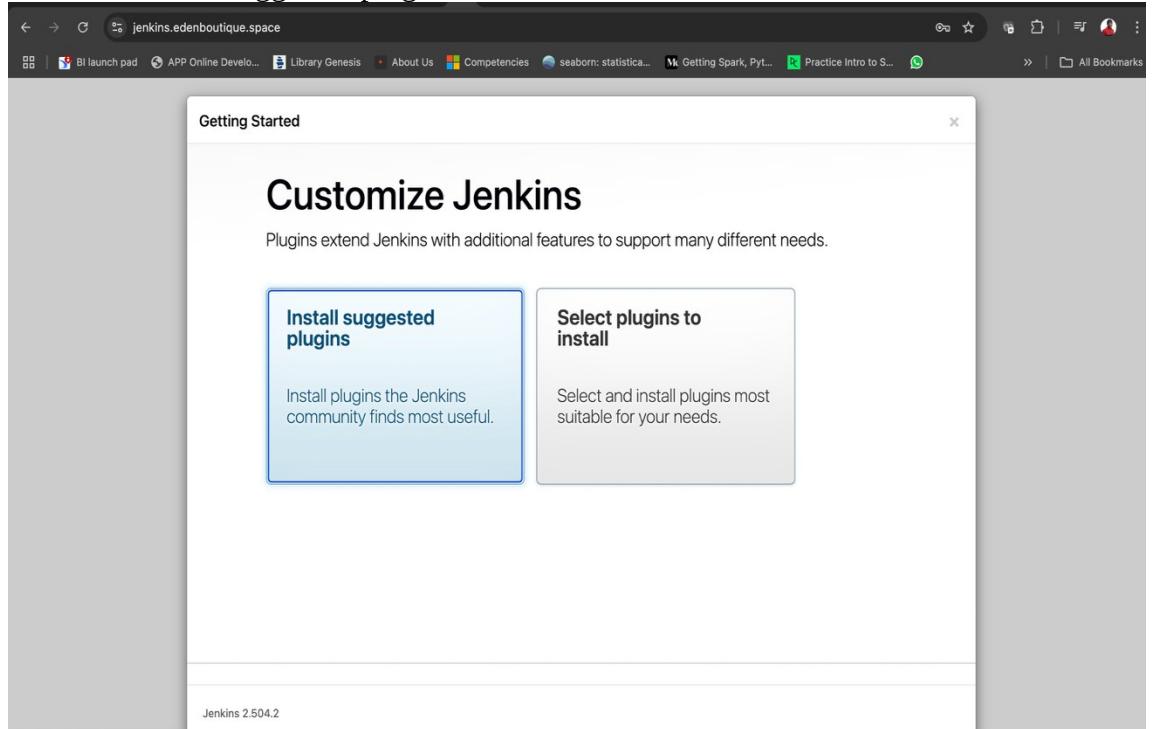
Administrator password

cat into the file path to copy the password

```
using rhc at https://red.ht/registration
[ec2-user@jenkins ~]$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
154139e9cd8a477481c03511862f653d
[ec2-user@jenkins ~]$
```

onyiafranklin (1 week ago)

paste then you would be prompted to install suggested plugins
click on “install suggested plugins”



Getting Started

Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	⌚ Credentials Binding	** Ionicons API Folders OWASP Markup Formatter ** ASM API ** JSON Path API ** Structs ** Pipeline: Step API ** Token Macro Build Timeout ** bouncycastle API
⌚ Timestamper	⌚ Workspace Cleanup	⌚ Ant	⌚ Gradle	
⌚ Pipeline	⌚ GitHub Branch Source	⌚ Pipeline: GitHub Groovy Libraries	⌚ Pipeline Graph View	
⌚ Git	⌚ SSH Build Agents	⌚ Matrix Authorization Strategy	⌚ PAM Authentication	
⌚ LDAP	⌚ Email Extension	⌚ Mailer	⌚ Dark Theme	

** - required dependency

Jenkins 2.504.2

fill in the blank spaces to create first admin user

Getting Started

Create First Admin User

Username

Password

Confirm password

Full name

E-mail address

Jenkins 2.504.2

[Skip and continue as admin](#)

[Save and Continue](#)

click on save and finish

Getting Started

Instance Configuration

Jenkins URL:

<https://jenkins.edenboutique.space/>

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.504.2

Not now

Save and Finish

click on “start using Jenkins”

Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

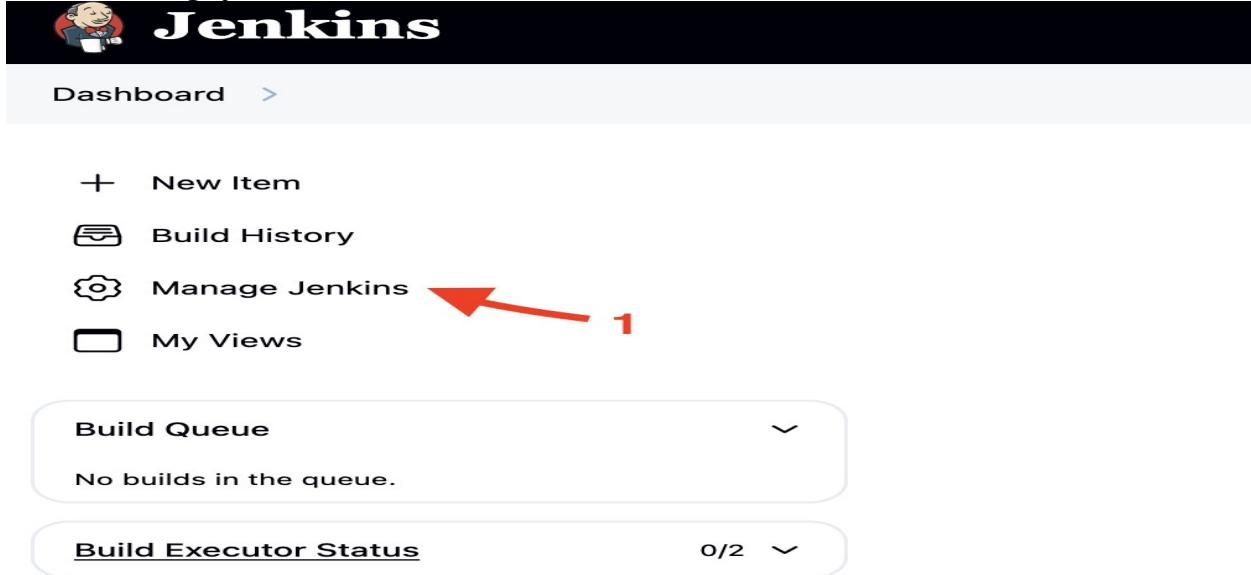
[Start using Jenkins](#)

Jenkins 2.504.2

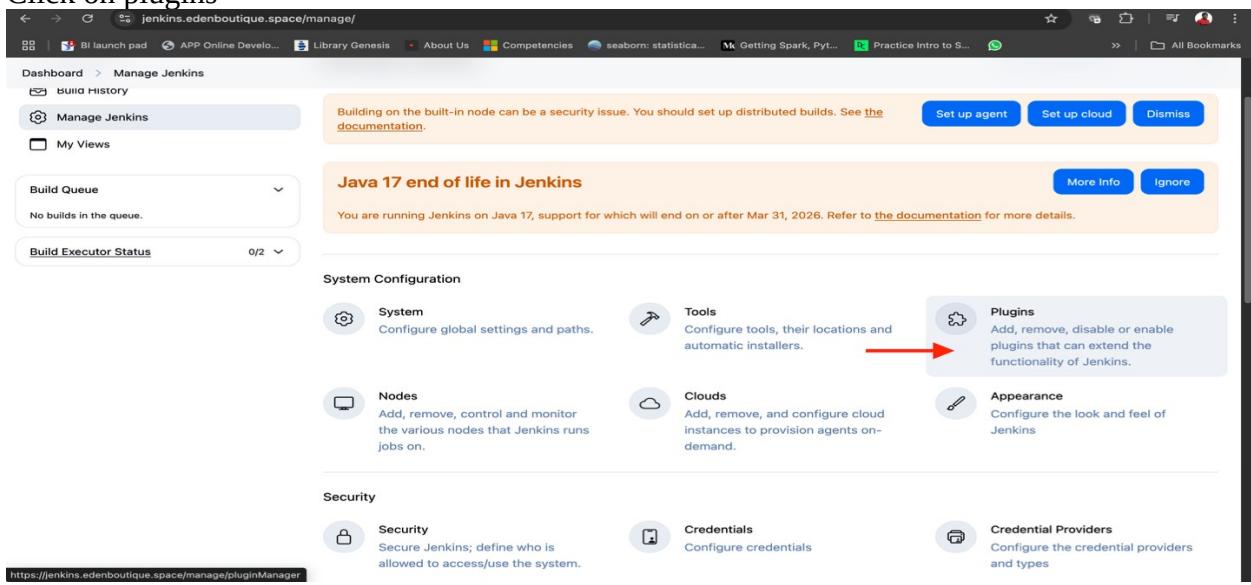
H. INFRASTRUCTURE SETUP AND PROVISIONING USING JENKINS PIPELINE

Here we would setup Jenkins to deploy infrastructure

click on manage jenkins



Click on plugins



Click on available plugins

A screenshot of a web browser displaying the Jenkins Manage Plugins page. The URL in the address bar is `jenkins.edenboutique.space/manage/plugins`. The page title is "Jenkins". The navigation path is "Dashboard > Manage Jenkins > Plugins". On the left, there is a sidebar with five options: "Updates", "Available plugins" (which is highlighted with a red arrow), "Installed plugins", "Advanced settings", and "Download progress". A search bar is located at the top right. The main content area is currently empty and labeled "Disabled row".

Plugins

Search

Updates

Available plugins

Installed plugins

Advanced settings

Download progress

Disabled row

Click on the search button

A screenshot of a web browser displaying the Jenkins plugin manager at jenkins.edenboutique.space/manage/pluginManager/available. The page title is "Jenkins". The left sidebar shows navigation options: "Updates", "Available plugins" (which is selected and highlighted in grey), "Installed plugins", "Advanced settings", and "Download progress". The main content area is titled "Plugins" and contains a search bar with the placeholder "Search available plugins" and a red arrow pointing to it. Below the search bar is a table listing several Jenkins plugins:

Install	Name	Released
<input type="checkbox"/>	JavaMail API 1.6.2-11	3 mo 7 days ago
<input type="checkbox"/>	Command Agent Launcher 123.v37cfcd92ef67	1 mo 20 days ago
<input type="checkbox"/>	Oracle Java SE Development Kit Installer 83.v417146707a_3d	4 mo 9 days ago
<input type="checkbox"/>	SSH server 3.353.v2b_d33c46e970	3 mo 19 days ago
<input type="checkbox"/>	Pipeline: REST API 2.38	1 mo 1 day ago
<input type="checkbox"/>	JSch dependency 0.2.16-95.v3eeeb_55fa_b_78	3 mo 4 days ago

Select terraform, AWS credentials, slack notification and Pipeline Stage View and click on install

A screenshot of a web browser displaying the Jenkins plugin manager at jenkins.edenboutique.space/manage/pluginManager/available. The page title is "Jenkins". The left sidebar shows navigation options: "Updates", "Available plugins" (selected), "Installed plugins", "Advanced settings", and "Download progress". The main content area is titled "Plugins" and contains a search bar with the placeholder "Search available plugins". Below the search bar is a table listing several Jenkins plugins, with three specific ones selected for installation (indicated by a checked checkbox in the "Install" column):

Install	Name	Released
<input checked="" type="checkbox"/>	Terraform 1.0.10	5 yr 3 mo ago
<input checked="" type="checkbox"/>	AWS Credentials 248.v78a_dcfc9db_ff	13 hr ago
<input checked="" type="checkbox"/>	Slack Notification 761.v2a_8770f0d169	4 mo 6 days ago
<input checked="" type="checkbox"/>	Pipeline: Stage View 2.38	1 mo 1 day ago

installation ongoing

Plugins

- Updates
- Available plugins
- Installed plugins
- Advanced settings
- Download progress**

Download progress

Preparation	
Ionicons API	Success
Folders	Success
OWASP Markup Formatter	Success
ASM API	Success
JSON Path API	Success
Structs	Success
Pipeline: Step API	Success
Token Macro	Success
Build Timeout	Success
bouncycastle API	Success
Credentials	Success
Plain Credentials	Success
Variant	Success
SSH Credentials	Success
Credentials Binding	Success
SCM API	Success
Pipeline: API	Success
commons-lang3 v3.x Jenkins API	Success

navigate to “manage jenkins” and click on credentials to add slack credentials

No builds in the queue.

You are running Jenkins on Java 17, support for which will end on or after Mar 31, 2026. Refer to [the documentation](#) for more details.

Build Executor Status 0/2

System Configuration

- System: Configure global settings and paths.
- Tools: Configure tools, their locations and automatic installers.
- Plugins: Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Nodes: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Clouds: Add, remove, and configure cloud instances to provision agents on-demand.
- Appearance: Configure the look and feel of Jenkins.

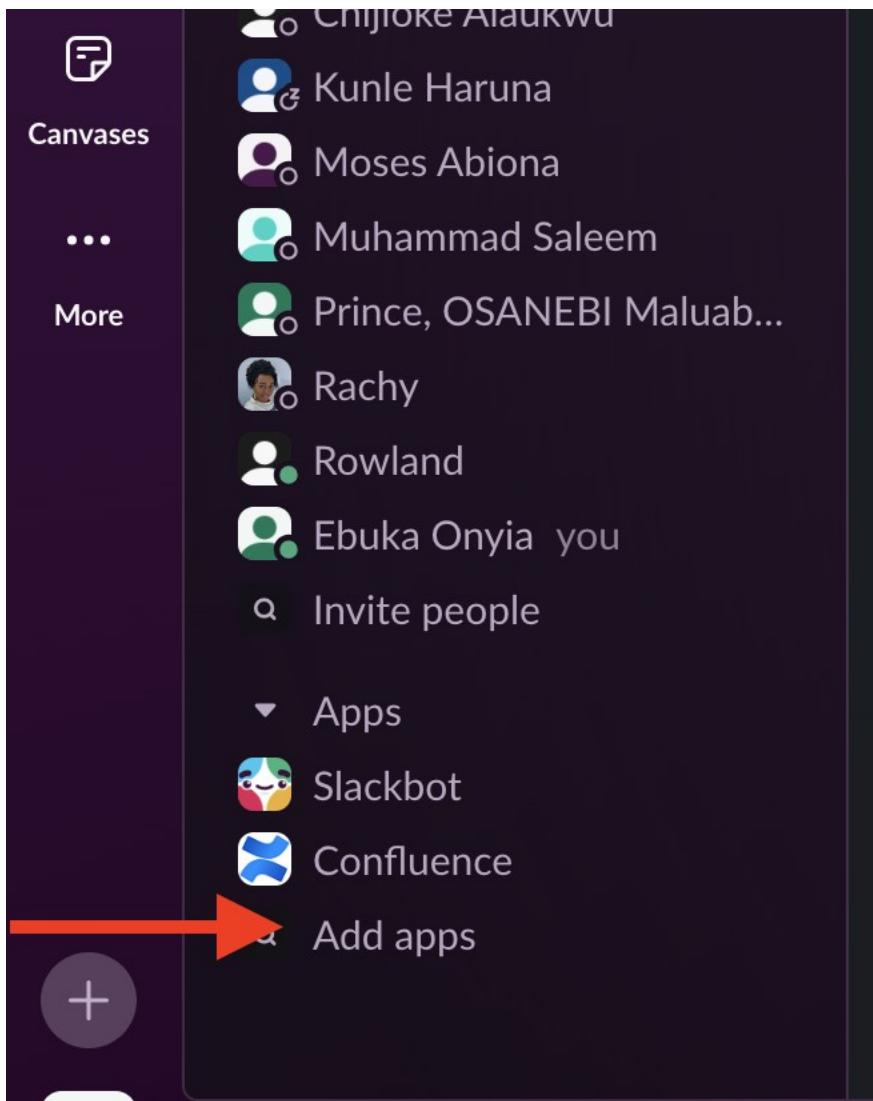
Security

- Security: Secure Jenkins; define who is allowed to access/use the system.
- Credentials**: Configure credentials (highlighted with a red arrow).
- Users: Create/delete/modify users that can log in to this Jenkins.
- Credential Providers: Configure the credential providers and types.

click on global

The screenshot shows a web browser window with the URL `jenkins.edenboutique.space/manage/credentials/`. The page title is "Jenkins". The navigation path is "Dashboard > Manage Jenkins > Credentials". The main section is titled "Credentials" and displays a table header with columns: T, P, Store ↓, Domain, and ID. Below this, a sub-section titled "Stores scoped to Jenkins" is shown, featuring a table with columns: P, Store ↓, Domains, and a "Domains" dropdown set to "(global)". A "System" entry is listed under the "Stores" column. To the right of the "Domains" dropdown, there is a button labeled "Add credentials" with a key icon, which is highlighted with a red arrow. At the bottom left of this section, there is an "Icon:" label followed by three letters: S, M, and L.

we need to generate a secret for our slack so we can add the credentials to slack. In other to get these secret/token, Go to your slack scroll down and click on “Add apps”



Click on Jenkins CI

A screenshot of the Slack Slackware interface. The sidebar on the left shows the following categories:

- Home
- DMS
- Activity
- Canvases
- More

The main area is titled "Apps" and features a search bar at the top. Below the search bar is a promotional message:

Bring all your tools into Slack
On Slack, apps belong to you and your team. Install once and everyone can use them. Teams on the free version of Slack get a limited number of apps.
[Upgrade to a paid plan](#) to get as many as you want.

Below the message, there is a dropdown menu labeled "All app types" and a list of apps:

11 apps in Cloudbight

- Jenkins CI

An orange arrow points to the Jenkins CI app entry in the list.

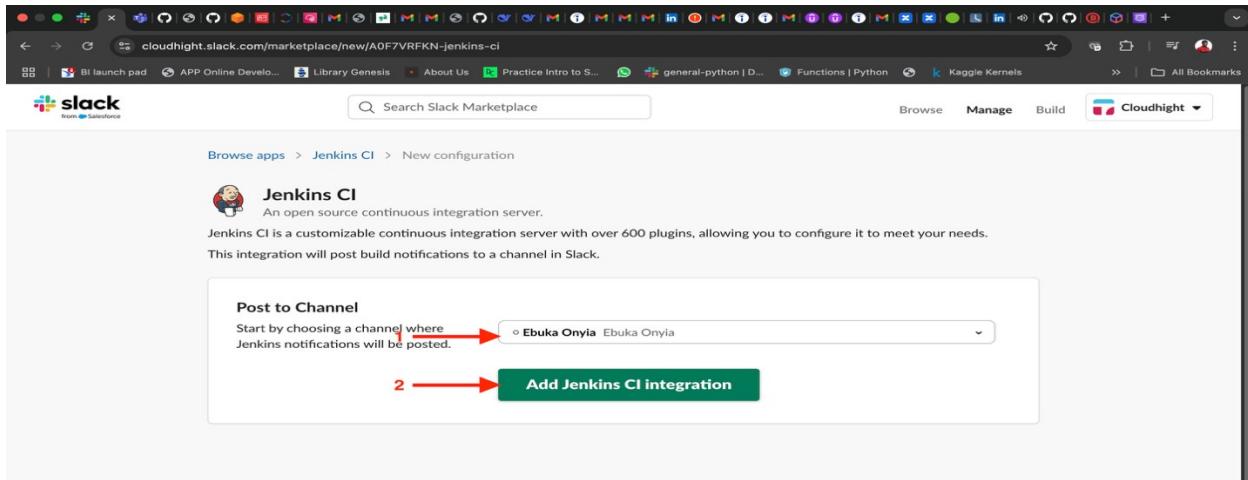
Click on Configuration

The screenshot shows the Slack interface for managing integrations. On the left, the sidebar includes Home, DMs, Activity, and Canvases. The main area displays the 'Jenkins CI' integration card. The card features a cartoon character holding a coffee cup, a title 'Jenkins CI', a subtitle 'Jenkins CI is a customizable needs.', and a 'Configuration' button. A red arrow points to the 'Configuration' button.

click on add to slack

The screenshot shows the Slack Marketplace page for the Jenkins CI integration. At the top, it says 'cloudhight.slack.com/marketplace/A0F7VRFKN-jenkins-ci'. The main content area shows the Jenkins CI app card with its logo, name, and a large 'Add to Slack' button. Below the button are links for 'Learn more & Support' and 'Privacy policy'.

Select the channel to post the notification. Here I selected my name and clicked on “Add Jenkins CI integration”



Scroll down to copy the token Credentials

Step 3

After it's installed, click on **Manage Jenkins** again in the left navigation, and then go to **Configure System**. Find the **Global Slack Notifier Settings** section and add the following values:

- Team Subdomain: `cloudhight`
- Integration Token Credential ID: Create a secret text credential using `xh2FVXd0121AcXrEE9NDh4tW` as the value

The other fields are optional. You can click on the question mark icons next to them for more information. Press **Save** when you're done.

Note: Please remember to replace the Integration Token in the screenshot below with your own.

Paste the token in the “secret tab” and give your slack credential a name, later, click on. create

New credentials

Kind: Secret text

Scope: Global (Jenkins, nodes, items, all child items, etc)

Secret: `.....`

ID: slack

Description: slack

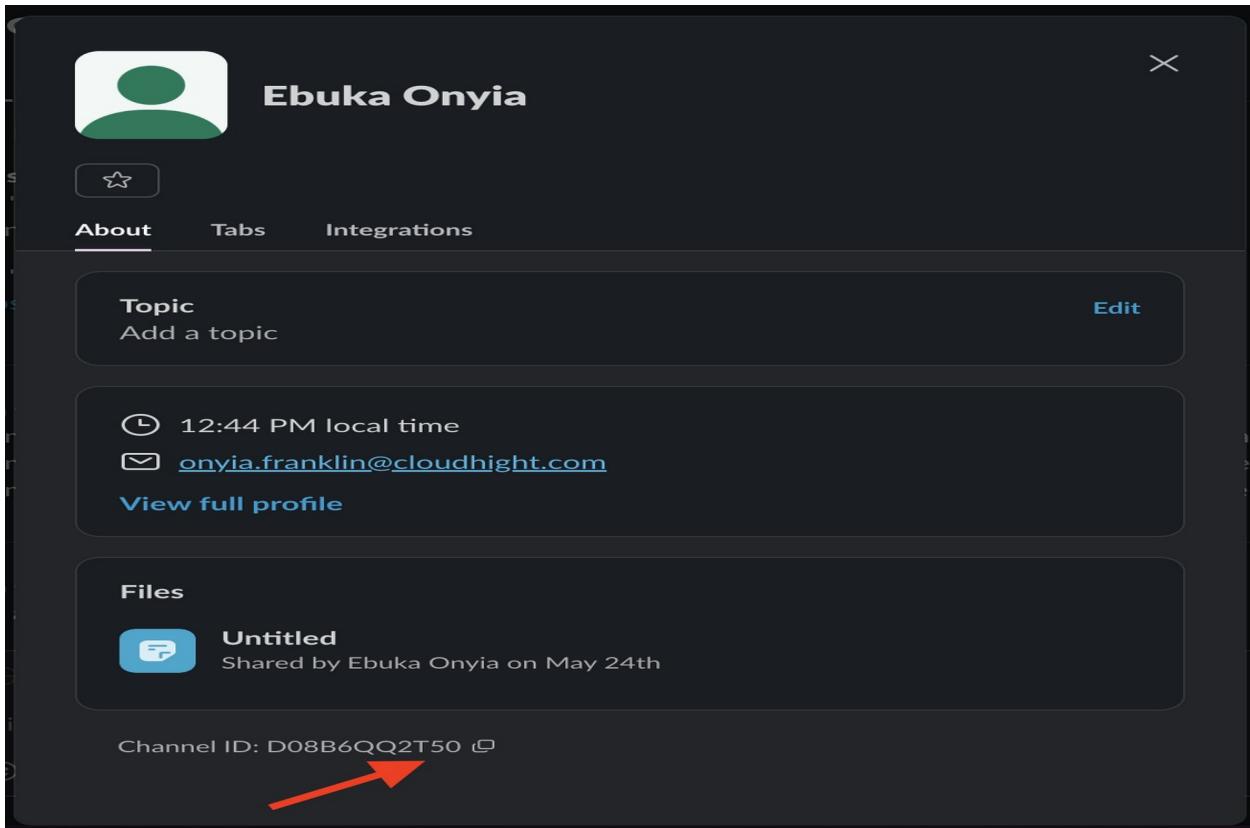
Create

The screenshot shows the Jenkins Global credentials (unrestricted) page. At the top, there's a navigation bar with links like Dashboard, Manage Jenkins, Credentials, System, and Global credentials (unrestricted). Below the navigation is a table header with columns: ID, Name, Kind, and Description. A single row is listed: ID is slack, Name is slack, Kind is Secret text, and Description is slack. There's also a blue 'Add Credentials' button at the top right.

Next is to set the slack as a tool. Navigate to manage Jenkins and click on tools.

The screenshot shows the Jenkins Manage Jenkins page. On the left, there's a sidebar with options: New Item, Build History, Manage Jenkins (which is highlighted), and My Views. The main area has a heading 'Manage Jenkins'. It features a warning box about Java 17 end-of-life and another box about Java 17 support. Below these are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (0/2). At the bottom, there's a 'System Configuration' section with a 'System' link.

scroll down to slack. Add workspace where you like the notification to be and test connection. Because we set our personal channel for the notification, visit your personal channel and copy the channel ID before adding it to the channel id space on Jenkins.



Click on test connection. You should get notification on your slack.
click on save and apply.

Slack

Workspace ?
Cloudhight

Credential ?
slack

+ Add

Default channel / member id ?
D08B6QQ2T50

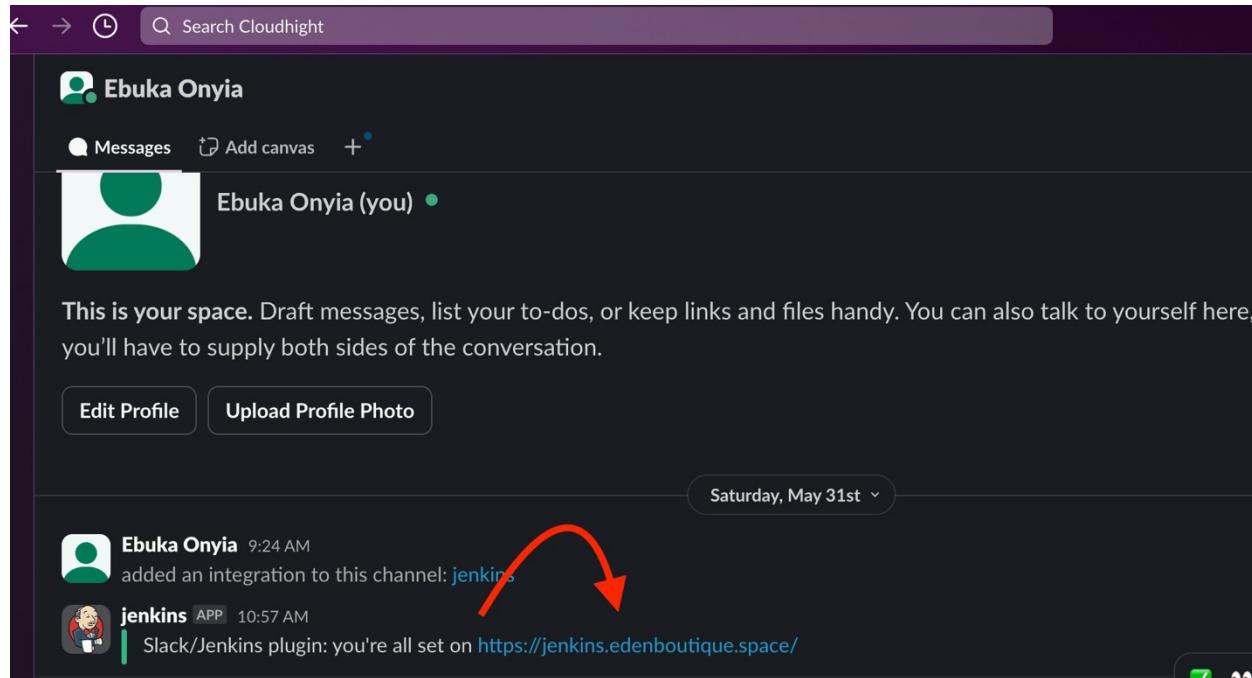
Custom slack app bot user ?

Advanced ▾

Success

Test Connection

Save Apply



Now lets add the github credentials

Select username with password

enter the username of your github

Generate a classic token from your github account and enter the token in the password.

Do this by visiting your github account and Navigate to settings> developers settings>personal access token>token classic> give the token a name and the generate it

A screenshot of a web browser showing the Jenkins "New credentials" configuration page. The URL is "jenkins.edenboutique.space/manage(credentials)/createUnrestrictedSystemDomain/_newCredentials". The page has a header with the Jenkins logo and navigation links like "Dashboard", "Manage Jenkins", "Credentials", "System", "Global credentials (unrestricted)", and "Create". The main form is titled "New credentials" and has a "Kind" dropdown set to "Username with password". It includes fields for "Scope" (set to "Global (Jenkins, nodes, items, all child items, etc)"), "Username" (set to "onyiafranklin"), "Password" (a masked input), "ID" (set to "git-cred"), and "Description" (set to "git-cred"). A "Create" button is at the bottom of the form.

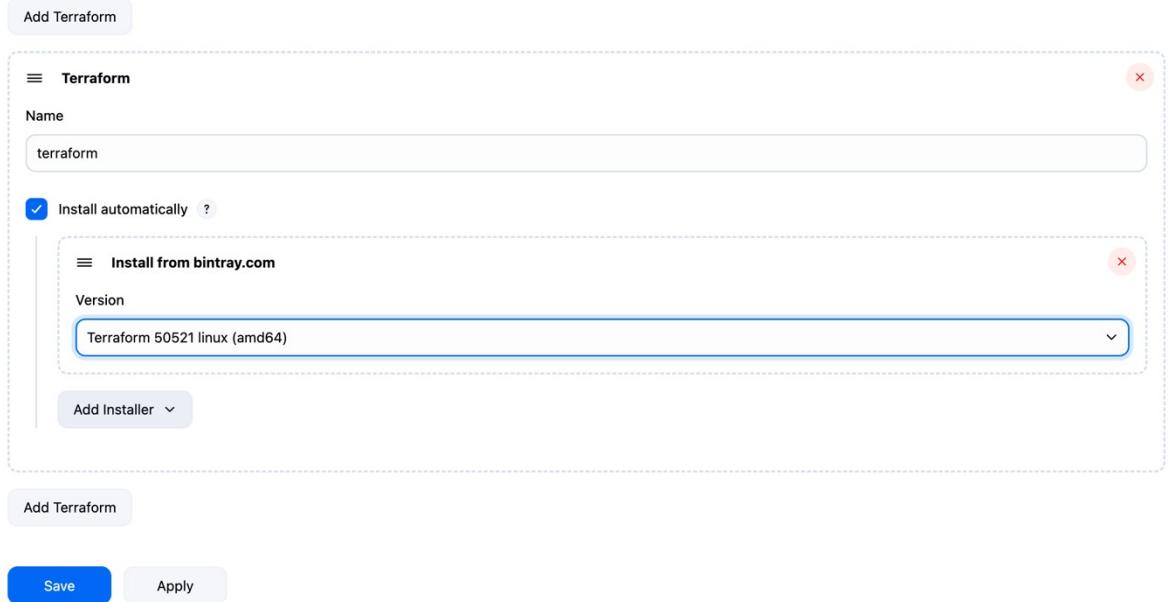
next , g back to tools on manage Jenkins , we want to install terraform tool

The screenshot shows the Jenkins Manage Jenkins interface. At the top, there is a banner about Java 17 end-of-life: "Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation." Below this, there are tabs for "Dashboard", "Manage Jenkins", and "Tools". The "Manage Jenkins" tab is selected. In the center, there is a "Java 17 end of life in Jenkins" message: "You are running Jenkins on Java 17, support for which will end on or after Mar 31, 2026. Refer to the documentation for more details." To the right of this message are buttons for "More Info" and "Ignore". Below the message, the "System Configuration" section is visible, containing links for "System", "Tools", "Nodes", "Clouds", and "Plugins".

scroll down and go to terraform installations and click on add terraform

The screenshot shows the Jenkins Tools configuration page. The URL in the browser is "jenkins.edenboutique.space/manage/configureTools/". The page has a breadcrumb navigation: "Dashboard > Manage Jenkins > Tools". The main content area is titled "Terraform installations" and contains a "Add Terraform" button. A red arrow points to this "Add Terraform" button. Below it are buttons for "Save" and "Apply".

Select install automatically and select linux (amd64)



Now we can create our infrastructure pipeline
Go to manage Jenkins and click on “new item”

The Jenkins Manage Jenkins page shows the following:

- + New Item** (highlighted with a red arrow)
- Manage Jenkins** (selected in the sidebar)
- System Configuration** section with links to:
 - System**: Configure global settings and paths.
 - Tools**: Configure tools, the automatic installers
 - Nodes**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
 - Clouds**: Add, remove, and configure cloud instances to provision demand.
- Java 17 end of life in Jenkins**: A warning message: "Building on the built-in node can be a security issue. You should set up distributed documentation." and "You are running Jenkins on Java 17, support for which will end on or after Mar 31, 2024."
- Build Queue**: No builds in the queue.
- Build Executor Status**: 0/2

<https://jenkins.edenboutique.space/view/all/new.job>
Give the pipeline a name and click ok

New Item

Enter an item name
infra-pipeline

Select an item type

Pipeline Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Freestyle project Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Multi-configuration project Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK

Choose “pipeline script from SCM”

Configure

General

Triggers

Pipeline

Advanced

Throttle builds ?

Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

Build after other projects are built ?
 Build periodically ?
 GitHub hook trigger for GITScm polling ?
 Poll SCM ?
 Trigger builds remotely (e.g., from scripts) ?

Pipeline

Define your Pipeline using Groovy directly or pull it from source control.

Definition

Pipeline script from SCM

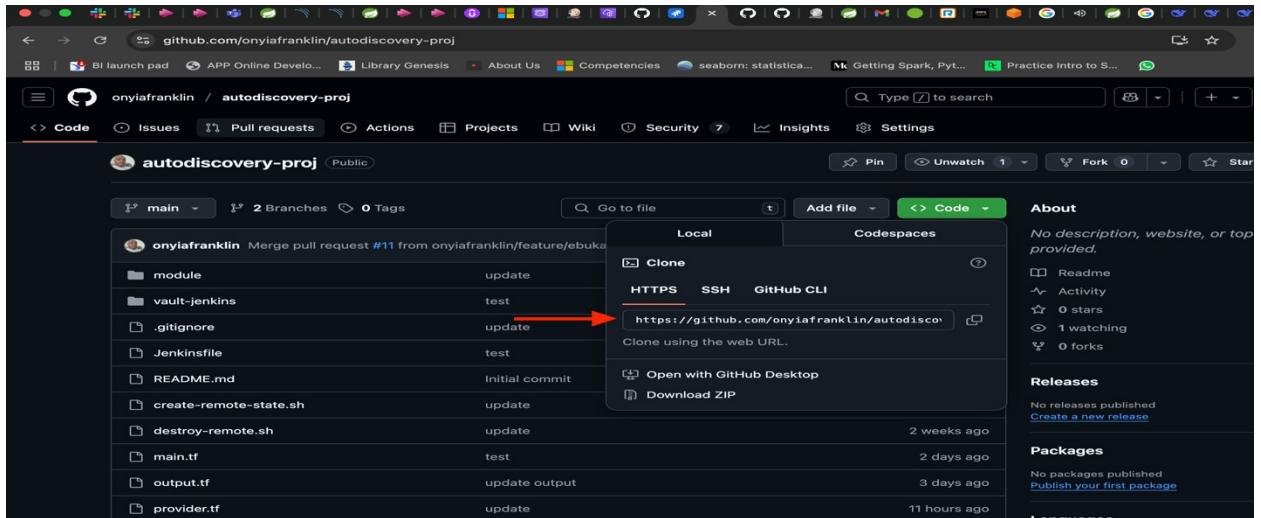
SCM ?

None

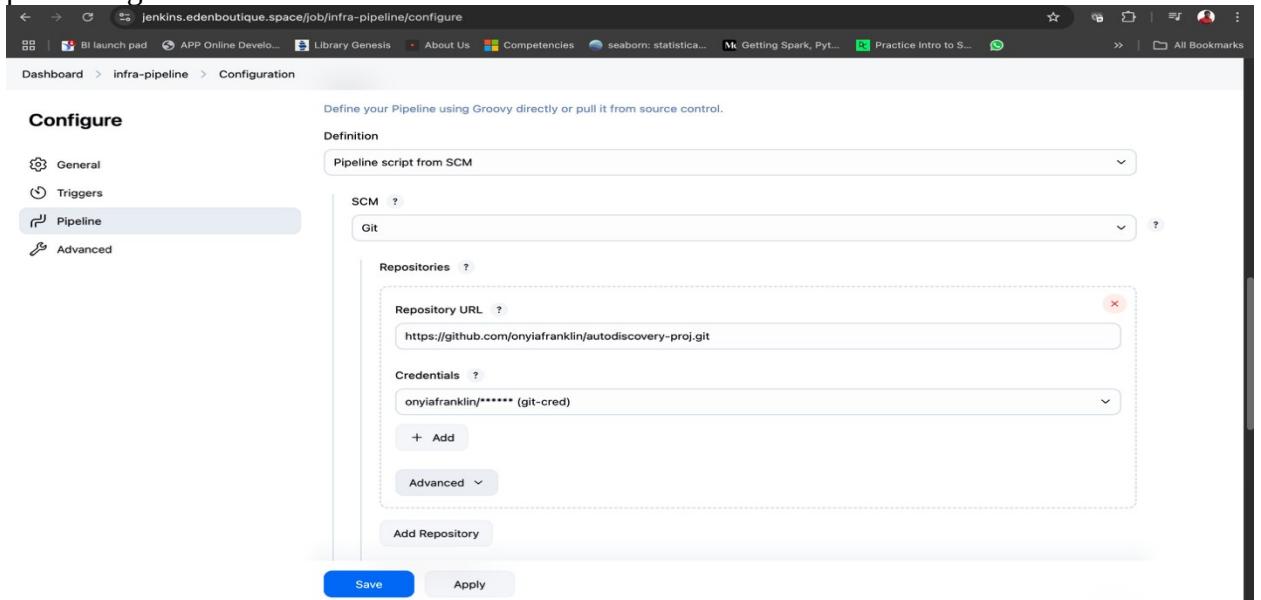
Script Path ?

Save Add

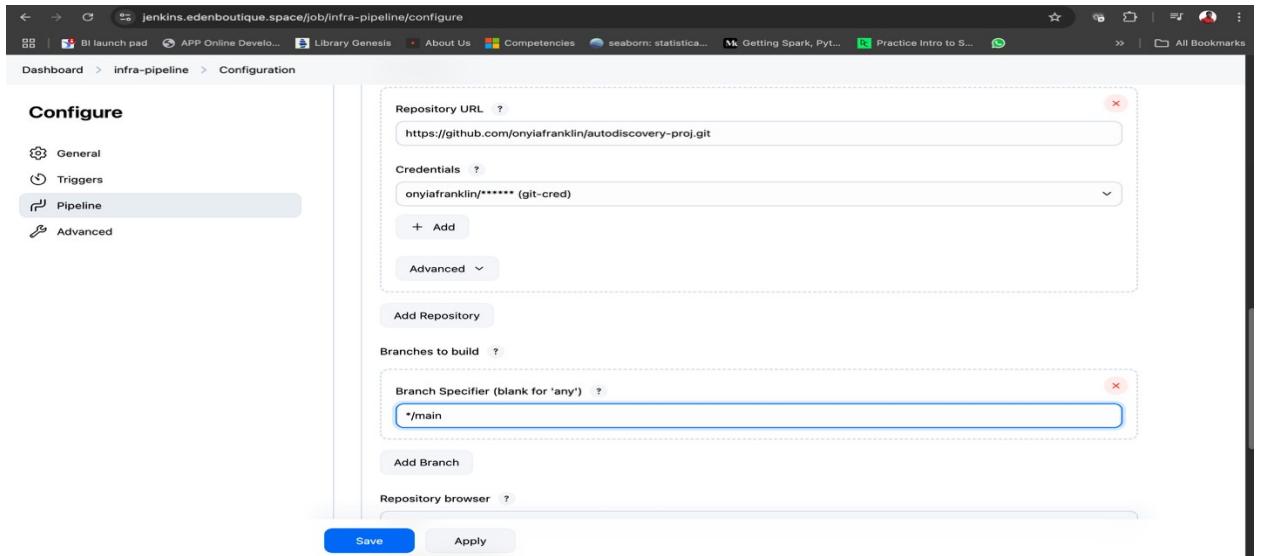
Copy your github url



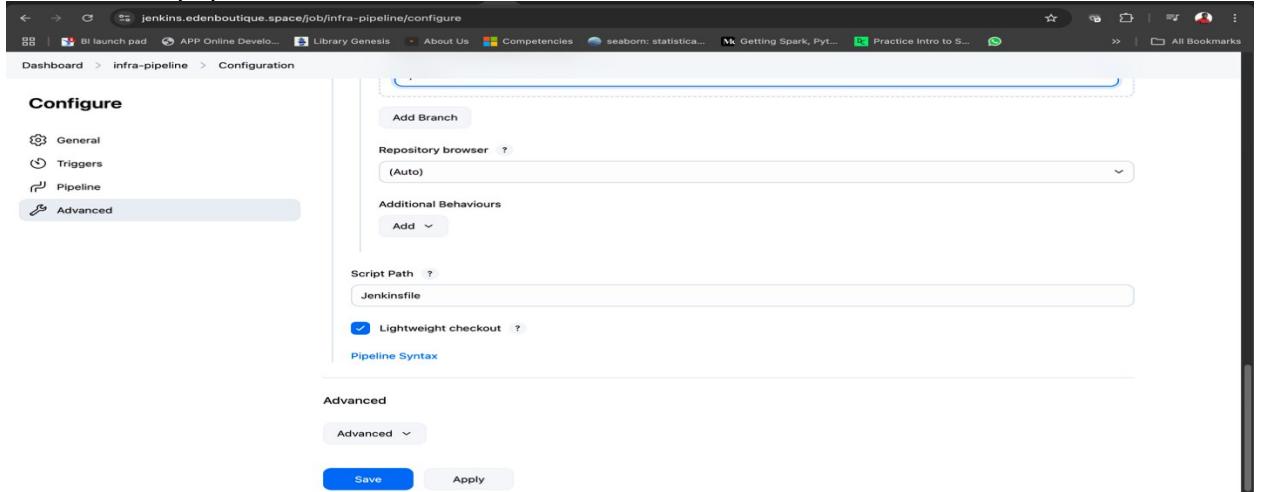
paste the github url



Select the main branch where I pushed the code



Enter the script path name “Jenkinsfile” and click ok



next is to update the vault token that would be used to unseal the vault
copy the generated token

A terminal window titled 'auto-discovery-mayist' is shown. The command 'cat token.txt' has been run, and the output is displayed. A red arrow labeled '1' points to the file 'token.txt' in the file explorer on the left. Another red arrow labeled '2' points to the copied vault token 's.arrE5QniMMk2EMPGpnpkKhWx' in the terminal output.

```
$ stage-bashscript.sh .../ansible
autodiscovery-proj > vault-jenkins > token.txt
1 s.arrE5QniMMk2EMPGpnpkKhWx
```

Paste the token in the root provider.tf under the vault provider

```

provider "aws" {
  region = "eu-west-2"
  # profile = "bukky_int"
}

provider "vault" {
  address = "https://vault.edenboutique.space"
  token   = "s.arrE50nIMk2EMPGpnpkKhWx"
}

terraform {
  backend "s3" {
    bucket      = "bucket-pet-adoption"
    key         = "infrastructure/terraform.tfstate"
    region     = "eu-west-2"
    use_lockfile = true
    encrypt    = true
  }
}

```

Commit and push back the code to repository

```

Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 313 bytes | 313.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/onyiafranklin/autodiscovery-proj.git
  689f8eb..59dc18d feature/ebuka --> feature/ebuka
● franklinonyia@MACBOOKPRO-CEA1 autodiscovery-proj % git status
On branch feature/ebuka
Your branch is up to date with 'origin/feature/ebuka'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified: provider.tf

no changes added to commit (use "git add" and/or "git commit -a")
● franklinonyia@MACBOOKPRO-CEA1 autodiscovery-proj % git add .
● franklinonyia@MACBOOKPRO-CEA1 autodiscovery-proj % git commit -m "update"
[feature/ebuka dac384c] update
  1 file changed, 1 insertion(+), 1 deletion(-)
○ franklinonyia@MACBOOKPRO-CEA1 autodiscovery-proj % ■

```



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ANSIBLE
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified: provider.tf

no changes added to commit (use "git add" and/or "git commit -a")
● franklinonyia@MACBOOKPRO-CEA1 autodiscovery-proj % git add .
● franklinonyia@MACBOOKPRO-CEA1 autodiscovery-proj % git commit -m "update"
[feature/ebuka dac384c] update
  1 file changed, 1 insertion(+), 1 deletion(-)
● franklinonyia@MACBOOKPRO-CEA1 autodiscovery-proj % git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 310 bytes | 310.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/onyiafranklin/autodiscovery-proj.git
  59dc18d..dac384c feature/ebuka --> feature/ebuka
○ franklinonyia@MACBOOKPRO-CEA1 autodiscovery-proj % ■

```

Go back to the Jenkins pipeline, refresh the url and click on “Build with parameters” select “build”

The screenshot shows the Jenkins Pipeline configuration page for a pipeline named 'infra-pipeline'. On the left, there's a sidebar with various options: Status, Changes, Build with Parameters, Configure (which is selected and highlighted in grey), Delete Pipeline, Full Stage View, Stages, Rename, and Pipeline Syntax. In the main area, the title is 'Pipeline infra-pipeline'. It says 'This build requires parameters:' followed by a 'action' dropdown menu with 'apply' selected. Below that are two buttons: 'Build' (green) and 'Cancel'. At the bottom, there's a 'Builds' section showing one build entry: '#1 10:14 AM'.

view the infrastructure provisioning progress

The screenshot shows the Jenkins Pipeline console output for build #2. The log starts with some pipeline script steps like [Pipeline] script, [Pipeline] slackSend, and [Pipeline] slackSend. It then shows two Slack Send Pipeline steps running with specific values. Following these are several [Pipeline] steps including slackSend, stage, withEnv, withCredentials, and withEnv again. The log concludes with '[Pipeline] End of Pipeline' and 'Finished: SUCCESS'.

```
[Pipeline] script
[Pipeline] {
[Pipeline] slackSend
Slack Send Pipeline step running, values are - baseUrl: <empty>, teamDomain: Cloudheight, channel: D08B6QQ2T50, color: good, botUser: false, tokenCredentialId: slack, notifyCommitters: false, iconEmoji: <empty>, username: <empty>, timestamp: <empty>
[Pipeline] }
[Pipeline] // script
[Pipeline] slackSend
Slack Send Pipeline step running, values are - baseUrl: <empty>, teamDomain: Cloudheight, channel: D08B6QQ2T50, color: good, botUser: false, tokenCredentialId: slack, notifyCommitters: false, iconEmoji: <empty>, username: <empty>, timestamp: <empty>
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Infrastructure successfully deployed

The screenshot shows the Jenkins Infra-Pipeline status page. On the left, there's a sidebar with options like Status, Changes, Build with Parameters, Configure, Delete Pipeline, Full Stage View, Stages, Rename, and Pipeline Syntax. Below that is a Builds section with a filter and two builds listed: #2 (10:17 AM) and #1 (10:14 AM). The main area is titled "infra-pipeline" and shows a timeline of stages: Declarative: Checkout SCM (600ms), Declarative: Tool Install (800ms), IAC Scan (20s), Terraform Init (5s), Terraform format (303ms), Terraform validate (3s), Terraform plan (3s), Terraform action (1min 55s), and Declarative: Post Actions (455ms). Below the timeline is a table of stage times:

Stage	Time
Declarative: Checkout SCM	600ms
Declarative: Tool Install	800ms
IAC Scan	20s
Terraform Init	5s
Terraform format	303ms
Terraform validate	3s
Terraform plan	3s
Terraform action	1min 55s
Declarative: Post Actions	455ms

Below the table, there's a "Permalinks" section with links to the last build and stable build.

View some resources created using terraform on console
here is the lunch template created for bastion, prod and stage servers

The screenshot shows the AWS EC2 Launch Templates page. The left sidebar includes EC2 (Dashboard, EC2 Global View, Events, Instances, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations), Images (AMIs, AMI Catalog), and a search bar. The main area displays "Launch Templates (3)" with a table:

Launch Template ID	Launch Template Name	Default Version	Latest Version	Create Time
lt-0fc647411af43b992	auto-discov-bastion-tmpl20250531101852...	1	1	2025-05-31T10:18:52.000Z
lt-0b71df311878e065c	auto-discov-prod-web-tmpl202505311019...	1	1	2025-05-31T10:19:07.000Z
lt-0c20f252ad73c1df2	auto-discov-stage-web-tmpl20250531101...	1	1	2025-05-31T10:19:07.000Z

Below the table is a "Select a launch template" section.

View all servers provisioned(prod,nexus,Jenkins,bastion,sonarqube, prod and stage all running)

eu-west-2.console.aws.amazon.com/ec2/home?region=eu-west-2#Instances:v=3;\$case:tags:true%5C,client:false,\$regex:tags:false%5C,client:false

EC2 Instances

Instances (1/8) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
auto-discov-prod-asg	i-05c1c88fa8532f4d0	Running	t2.medium	2/2 checks passed	View alarm
auto-discov-nexus-server	i-0b7bcc15243c47581	Running	t2.medium	2/2 checks passed	View alarm
auto-discov-jenkins-server	i-0cce9915d54b1d21	Running	t2.medium	2/2 checks passed	View alarm
auto-discov-VaultServer	i-0f62358e1a78b6e9f	Running	t2.medium	2/2 checks passed	View alarm
auto-discov-ansible-server	i-0ff9cd756d34c84de	Running	t2.medium	2/2 checks passed	View alarm
auto-discov-bastion-asg	i-086d3a0cd1e09b20	Running	t2.medium	2/2 checks passed	View alarm
auto-discov-sonarqube-server	i-0ed35228cc23953c7	Running	t2.medium	2/2 checks passed	View alarm
auto-discov-stage-asg	i-05ac184027b08b46f	Running	t2.medium	2/2 checks passed	View alarm

i-0b7bcc15243c47581 (auto-discov-nexus-server)

view elastic ip created

eu-west-2.console.aws.amazon.com/ec2/home?region=eu-west-2#Addresses:

Elastic IP addresses (1) Info

Name	Allocated IPv4 address	Type	Allocation ID	Reverse DNS record
auto-discov-eip	3.9.219.244	Public IP	eipalloc-08d27c26ba3236eac	-

Select an elastic IP address

view all load balancers created for vault,Jenkins,nexus,sonarqube, prod and stage servers

eu-west-2.console.aws.amazon.com/ec2/home?region=eu-west-2#LoadBalancers:

Load balancers (6)

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Name	DNS name	State	VPC ID	Availability Zones	Type
auto-discov-elb-sonar	auto-discov-elb-sonar-7236...	-	vpc-036c4cf246e65b96a	2 Availability Zones	classic
elb-jenkins	elb-jenkins-1163844456.eu...	-	vpc-0dc620d69ea05bcc9	2 Availability Zones	classic
auto-discov-nexus-elb	auto-discov-nexus-elb-9845...	-	vpc-036c4cf246e65b96a	2 Availability Zones	classic
vault-elb	vault-elb-1217819938.eu...	-	vpc-0dc620d69ea05bcc9	2 Availability Zones	classic
auto-discov-prod-LB	auto-discov-prod-LB-12489...	Active	vpc-036c4cf246e65b96a	2 Availability Zones	application
auto-discov-stage-LB	auto-discov-stage-LB-2035...	Active	vpc-036c4cf246e65b96a	2 Availability Zones	application

0 load balancers selected

Select a load balancer above.

see auto scaling groups created for bastion, stage and prod servers

The screenshot shows the AWS EC2 Auto Scaling Groups page. The left sidebar includes sections for Images, Elastic Block Store, Network & Security, Load Balancing, and Auto Scaling. Under Auto Scaling, 'Auto Scaling Groups' is selected. The main content area displays a table titled 'Auto Scaling groups (3)'. The table has columns for Name, Launch template/configuration, Instances, Status, Desired capacity, and Min. The three listed groups are:

Name	Launch template/configuration	Instances	Status	Desired capacity	Min
auto-discov-prod-asg	auto-discov-prod-web-tmpl2025053110	1	-	1	1
auto-discov-stage-asg	auto-discov-stage-web-tmpl2025053110	1	-	1	1
auto-discov-bastion-asg	auto-discov-bastion-tmpl202505311018	1	-	1	1

view the vpc provisioned

The screenshot shows the AWS VPC Your VPCs page. The left sidebar includes sections for VPC dashboard, Virtual private cloud, and Subnets. Under Virtual private cloud, 'Your VPCs' is selected. The main content area displays a table titled 'Your VPCs (2)'. The table has columns for Name, VPC ID, State, Block Public..., IPv4 CIDR, and IPv6 CIDR. The listed VPCs are:

Name	VPC ID	State	Block Public...	IPv4 CIDR	IPv6 CIDR
auto-discov-vpc	vpc-036c4cf246e65b96a	Available	Off	10.0.0.0/16	-
-	vpc-0dc620d69ea05bcc9	Available	Off	172.31.0.0/16	-

view the subnets provisioned

Subnets (7) Info

Last updated less than a minute ago

Actions | Create subnet

Name	Subnet ID	State	VPC	Block Public...
-	subnet-047ff3b67749eabd	Available	vpc-0dc620d69ea05bcc9	Off
auto-discov-pri_sub2	subnet-0c3ca2b6dce711858	Available	vpc-036c4cf246e65b96a auto...	Off
auto-discov-pub_sub1	subnet-0031203d6b59149ae	Available	vpc-036c4cf246e65b96a auto...	Off
-	subnet-0b485673a52b7b5ce	Available	vpc-0dc620d69ea05bcc9	Off
auto-discov-pri_sub1	subnet-049380aa5fbe17fb4	Available	vpc-036c4cf246e65b96a auto...	Off
auto-discov-pub_sub2	subnet-06f48bc4f806cd9	Available	vpc-036c4cf246e65b96a auto...	Off
-	subnet-08817350eaec8236e9	Available	vpc-0dc620d69ea05bcc9	Off

Select a subnet

view route tables provisioned

Route tables (4) Info

Last updated less than a minute ago

Actions | Create route table

Name	Route table ID	Explicit subnet associ...	Edge associations	Main	VPC
auto-discov-pub_rt	rtb-0f8929ff116579f7c	2 subnets	-	No	vpc-036c4cf246
auto-discov-pri_rt	rtb-0dc1818f69009893e	2 subnets	-	No	vpc-036c4cf246
-	rtb-01a8861bf836db0e2	-	-	Yes	vpc-0dc620d69r
-	rtb-043be730a118f69e9	-	-	Yes	vpc-036c4cf246

Select a route table

view internet gateway

Internet gateways (2) Info

Last updated less than a minute ago

Actions | Create internet gateway

Name	Internet gateway ID	State	VPC ID	Owner
auto-discov-igw	igw-06a6e8e070e5cbf9	Attached	vpc-036c4cf246e65b96a auto-discov-...	896035:
-	igw-0ae9330897f05fc6d	Attached	vpc-0dc620d69ea05bcc9	896035:

view natgateway

Name	NAT gateway ID	Connectivity...	State	State message	Primary pub...
auto-discov-nwg	nat-0e94648d7407ca460	Public	Available	-	3.9.219.244

view security groups provisioned

Name	Security group ID	Security group name	VPC ID
-	sg-0a69495904e696cc3	default	vpc-0dc620d69ea05bcc9
auto-discov-nexus-sg	sg-0sec7c2bd0508a69b	auto-discov-nexus-sg	vpc-036c4cf246e65b96a
-	sg-020646322c8bc9c11	auto-discov-jenkins-elb-sg	vpc-0dc620d69ea05bcc9
-	sg-0f0dffaa8e6c3f18f5	default_elb_8ac4f631-9fe2-3edf-b9f3-...	vpc-0dc620d69ea05bcc9
auto-discov-prod-elb-sg	sg-0e660e0f62ef4e204	auto-discov-prod-elb-sg	vpc-036c4cf246e65b96a
auto-discov-stage-elb-sg	sg-0a30876f2fdffbb823	auto-discov-stage-elb-sg	vpc-036c4cf246e65b96a
auto-discov-stage-sg	sg-05bbf07537389dc46	auto-discov-stage-sg	vpc-036c4cf246e65b96a
auto-discov-rds-sg	sg-04678b96dcda6d6187	auto-discov-rds-sg	vpc-036c4cf246e65b96a
-	sg-02bb839c09deae778	default	vpc-036c4cf246e65b96a

Select a security group

view database provisioned

view scripts copied from the s3 bucket to ansible server by ssh into the ansible server

```
You can learn more about how to register your system
using rhc at https://red.ht/registration
Last login: Sat May 31 17:05:02 2025 from 10.0.1.198
[ec2-user@ansible-server ~]$ ls /etc/ansible
ansible.cfg      deployment.yml  prod.lists  stage-bashscript.sh  stage.lists
ansible_vars_file.yml  hosts      prod_hosts   roles          stage_hosts
[ec2-user@ansible-server ~]$ cat /etc/ansible/stage_hosts
[webservers]
10.0.4.220 ansible_user=ec2-user ansible_ssh_private_key_file=/home/ec2-user/.ssh/id_rsa
[ec2-user@ansible-server ~]$
```

check to see the content that would be copied inside the ansible host

```
10.0.4.220 ansible_user=ec2-user ansible_ssh_private_key_file=/home/ec2-user/.ssh/id_rsa
[ec2-user@ansible-server ~]$ cat /etc/ansible/prod_hosts
[webservers]
10.0.3.247 ansible_user=ec2-user ansible_ssh_private_key_file=/home/ec2-user/.ssh/id_rsa
[ec2-user@ansible-server ~]$
```

do same for stage servers

```
10.0.3.247
[ec2-user@ansible-server ~]$ cat /etc/ansible/stage.lists
10.0.4.220
[ec2-user@ansible-server ~]$
```

View results from quality gate scan

SonarQube dashboard for the petclinic project. The Quality Gate Status is **Passed** (All conditions passed). The dashboard displays various measures:

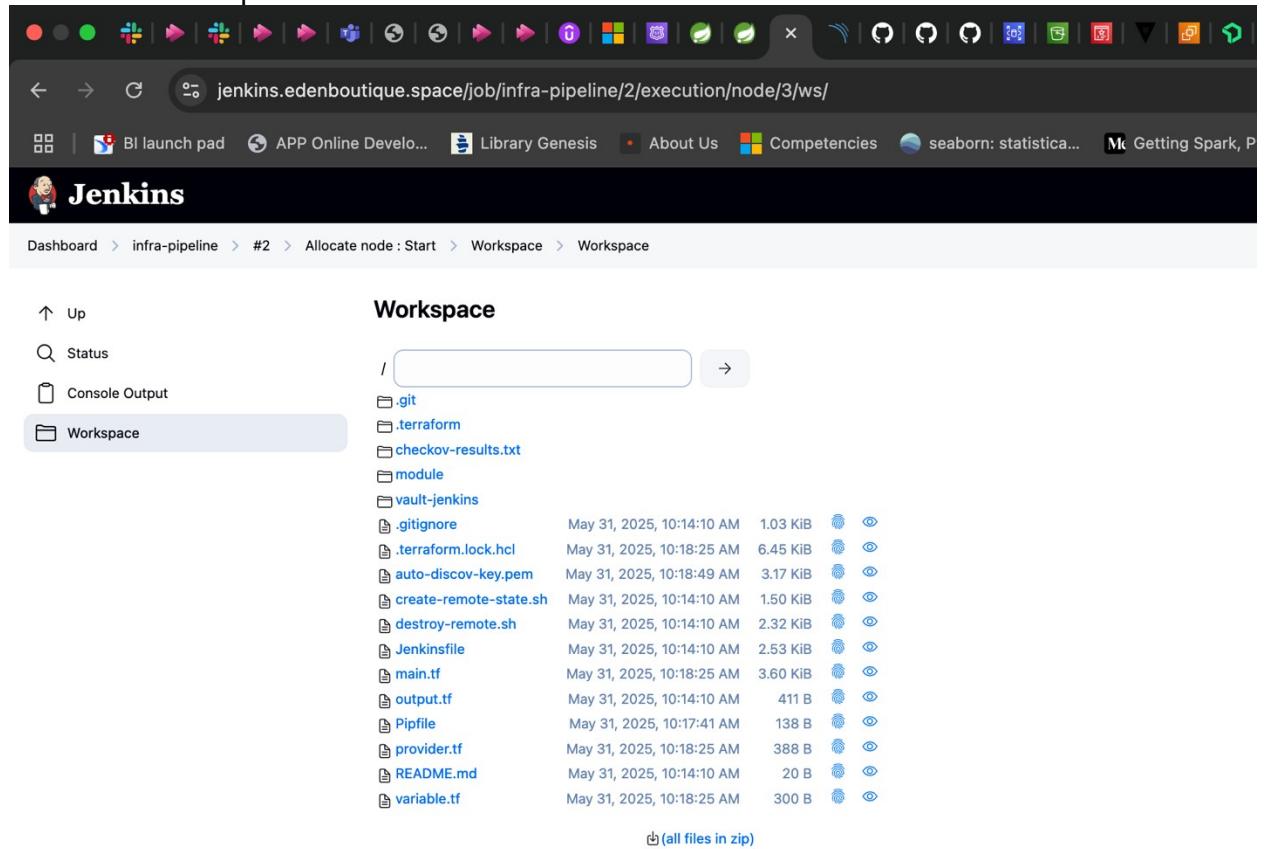
- MEASURES** section:
 - New Code: Started May 31, 2025, 7 hours ago.
 - Overall Code: Reliability (A)
 - New Bugs: Security (A)
 - New Vulnerabilities: Security Review (A)
 - New Security Hotspots: Reviewed
 - Added Debt: Maintainability (A)
 - New Code Smells: Duplications on 0 New Lines
 - Coverage on 0 New Lines to cover
 - Duplications on 0 New Lines

view results from dependency scan

Jenkins Dependency-Check Results page. The table shows the following vulnerabilities:

File Name	Vulnerability	Severity	Weakness
bootstrap-3.3.6.jar	NVD CVE-2016-10735	Medium	CWE-79
bootstrap-3.3.6.jar	NVD CVE-2018-14041	Medium	CWE-79
bootstrap-3.3.6.jar	NVD CVE-2018-14042	Medium	CWE-79
bootstrap-3.3.6.jar	NVD CVE-2018-20676	Medium	CWE-79
bootstrap-3.3.6.jar	OSSINDEX CVE-2018-14040	Medium	CWE-79
bootstrap-3.3.6.jar	OSSINDEX CVE-2018-20677	Medium	CWE-79
bootstrap-3.3.6.jar	OSSINDEX CVE-2019-8331	Medium	CWE-79
bootstrap-3.3.6.jar	OSSINDEX CVE-2024-6484	Medium	CWE-79
bootstrap-3.3.6.jar	OSSINDEX CVE-2024-6531	Medium	CWE-79
bootstrap-3.3.6.jar	OSSINDEX CVE-2024-6485	Low	CWE-79

see Jenkins workspace.



Next is to deploy our application on already deployed infrastructure
to do that, we need to deploy our application on a different pipeline.
lets first of all write our Jenkins and docker file that our Jenkins pipeline would run.

I. APPLICATION JENKINSFILE EXPLANATION

The application pipeline starts with declaration of pipeline that encloses the whole pipeline stages its environment and agent to run the pipeline. Because we have not specified any agent to run the pipeline, it uses any available server , which is the jenkins server to run this pipeline. next, we declared some environment variable that would be used in this pipeline

nexus user , nexus password an d nexus-docker-repo are credentials that jenkins would use to login to the docker nexus repo to run the docker image.

here is the explanation of the stages of the pipeline

Here's a detailed breakdown of your Jenkins pipeline stages with clear headings for each stage:

1. Code Analysis Stage

Tool: SonarQube

Purpose: Performs static code analysis to identify bugs, vulnerabilities, and code quality issues

Process:

- Executes `mvn sonar:sonar` command to scan the Maven project
- Integrates with pre-configured SonarQube server ('sonarqube')
- Generates detailed reports on code metrics (coverage, duplications, security issues)

2. Quality Gate

Tool: SonarQube Quality Gate

Purpose: Ensures code meets quality standards before proceeding

Process:

- Waits for SonarQube analysis results (timeout: 2 minutes)
- Automatically aborts pipeline if quality thresholds aren't met

3. Dependency Check

Tool: OWASP Dependency-Check + NVD API

Purpose: Identifies vulnerable dependencies in the project

Process:

- Scans project dependencies using NVD API key for vulnerability data .NB:NVD-key help the dependency scan process to be faster
- Generates report (`dependency-check-report.xml`)
- Explicitly disables Node/Yarn audits (focuses on Java dependencies)

4. Build Artefacts

Tool: Maven

Purpose: Compiles application into deployable package

Process:

- Runs `mvn clean package` with tests and checkstyle skipped on this stage, this is the life cycle of what happens

- . clean Phase (from the Clean Lifecycle)
 - Deletes the target/ directory (if it exists), removing all previously compiled files and artifacts.
- . validate (Default Lifecycle)
 - Checks if the project structure is correct (e.g., pom.xml exists).
- . compile
 - Compiles the main Java source code (src/main/java) into bytecode (stored in target/classes).
- . test-compile
 - Compiles test source code (src/test/java) into bytecode (stored in target/test-classes).
- . test
 - Runs unit tests (using a testing framework like JUnit) from src/test/java.
 - Fails the build if any test fails (unless skipped via -DskipTests).
- package
 - Packages the compiled code (target/classes) and resources (src/main/resources) into the specified packaging format (e.g., JAR, WAR, EAR).
 - For a JAR: Creates a .jar file in target/.
 - For a WAR: Creates a .war file for web applications.
 - Produces spring-petclinic-2.4.2.war in target directory

5. Push Artifacts to Nexus-Repo

Tool: Nexus Artifact Uploader

Purpose: Stores WAR file in Nexus repository

Process:

- Uploads WAR file to nexus.edenboutique.space via HTTPS
- Uses credentials ('nexus-cred') for authentication
- Organizes artifact with:
 - Group ID: Petclinic
 - Artifact ID: spring-petclinic
 - Version: 1.0

6. Build Docker Image

Tool: Docker

Purpose: Creates containerized version of application

Process:

- Builds image from Dockerfile using `docker build`
- Tags image as `$NEXUS_REPO/petclinicapps`

7. Login to Nexus Repo

Tool: Docker CLI

Purpose: Authenticates to private Nexus Docker registry

Process:

- Uses stored credentials (`NEXUS_USER`, `NEXUS_PASSWORD`)
- Executes `docker login` command

8. Push Image to Nexus Repo

Tool: Docker CLI

Purpose: Stores Docker image in Nexus for deployment

Process:

- Pushes the built image to Nexus repository
- Makes image available for deployment stages

9. Trivy Image Scan

Tool: Trivy

Purpose: Security scans the Docker image

Process:

- Analyzes image for OS/library vulnerabilities
- Outputs results to `trivy.txt` file

10. Deploy to Stage

Tool: Ansible + SSH Agent

Purpose: Deploys application to staging environment

Process:

- Uses SSH agent with 'ansible-key' credentials
- Connects through bastion host (`BASTION_IP`) to executes Ansible playbook (`deployment.yml`) against stage hosts

11. Check Stage Website Availability

Tool: cURL + Slack

Purpose: Verifies successful staging deployment

Process:

- Waits 200 seconds for application to initialize
- Checks HTTP status of stage.edenboutique.space
- Sends Slack notification with results:
 - Success (HTTP 200): Green message
 - Failure: Red alert message

12. Request for Approval

Tool: Jenkins Input Step

Purpose: Manual approval before production deployment

Process:

- Pauses pipeline for up to 10 minutes
- Requires approval from 'admin' role

13. Deploy to Prod

Tool: Ansible + SSH Agent

Purpose: Deploys to production environment

Process:

- Similar to stage deployment
- Uses production inventory (prod_hosts)
- Same Ansible playbook but targets production servers

14. Check Prod Website Availability

Tool: cURL + Slack

Purpose: Validates production deployment

Process:

- Checks www.edenboutique.space after 200s warmup
- Sends Slack notification with status
- Differentiates between stage/prod in messages

```
pipeline {
    agent any
    environment {
        NEXUS_USER = credentials('nexus-username')
        NEXUS_PASSWORD = credentials('nexus-password')
```

```

NEXUS_REPO = credentials('nexus-docker-repo')
NVD_API_KEY= credentials('nvd-key')
ANSIBLE_IP = credentials('ansible-ip')
BASTION_IP = credentials('bastion-ip')
}
stages {
    stage('Code analisys stage') {
        steps {
            withSonarQubeEnv('sonarqube') {
                sh 'mvn sonar:sonar'
            }
        }
    }
    stage('Quality gate') {
        steps {
            timeout(time: 2, unit: 'MINUTES') {
                waitForQualityGate abortPipeline: true
            }
        }
    }
    stage('Dependency check') {
        steps {
            withCredentials([string(credentialsId: 'nvd-key', variable: 'NVD_API_KEY')]) {
                dependencyCheck additionalArguments: "--scan ./ -- disableYarnAudit --disableNodeAudit --nvdApiKey $NVD_API_KEY",
                odcInstallation: 'DP-Check'
            }
            dependencyCheckPublisher pattern: '**/dependency-check-report.xml'
        }
    }
    stage('Build artefacts') {
        steps {
            sh 'mvn clean package -DskipTests -Dcheckstyle.skip'
        }
    }
    stage('Push artifacts to nexus-repo') {
        steps {
            nexusArtifactUploader artifacts: [[artifactId: 'spring-petclinic',
                classifier: '',
                file: 'target/spring-petclinic-2.4.2.war',
                type: 'war']],
            credentialsId: 'nexus-cred',
            groupId: 'Petclinic',
            nexusUrl: 'nexus.edenboutique.space',
            nexusVersion: 'nexus3',
            protocol: 'https',
            repository: 'nexus-repo',
            version: '1.0'
        }
    }
    stage('Build Docker image') {
        steps {

```

```

        sh 'docker build -t $NEXUS_REPO/petclinicapps .'
    }
}

stage('Login to Nexus repo') {
    steps {
        sh 'docker login --username $NEXUS_USER --password
$NEXUS_PASSWORD $NEXUS_REPO'
    }
}
stage('Push image to Nexus repo') {
    steps {
        sh 'docker push $NEXUS_REPO/petclinicapps'
    }
}
stage('Trivy image scan') {
    steps {
        sh "trivy image $NEXUS_REPO/petclinicapps > trivy.txt"
    }
}
stage('Deploy to stage') {
    steps {
        sshagent(['ansible-key']) {
            sh '''
                ssh -t -t -o StrictHostKeyChecking=no -o
ProxyCommand="ssh -W %h:%p -o StrictHostKeyChecking=no ec2-user@${
BASTION_IP} ec2-user@${ANSIBLE_IP} "ansible-playbook -i
/etc/ansible/stage_hosts /etc/ansible/deployment.yml"
            '''
        }
    }
}
stage('check stage website availability') {
    steps {
        sh "sleep 200"
        sh "curl -s -o /dev/null -w \"%{http_code}\"
https://stage.edenboutique.space"
        script {
            def response = sh(script: "curl -s -o /dev/null -w \"%
$http_code\" https://stage.edenboutique.space", returnStdout: true).trim()
            if (response == "200") {
                slackSend(color: 'good', message: "The stage
petclinic website is up and running with HTTP status code ${response}.",
tokenCredentialId: 'slack')
            } else {
                slackSend(color: 'danger', message: "The stage
petclinic wordpress website appears to be down with HTTP status code $(
$response).", tokenCredentialId: 'slack')
            }
        }
    }
}
stage('Request for Approval') {
    steps {
        timeout(activity: true, time: 10) {
            input message: 'Needs Approval ', submitter: 'admin'
        }
    }
}

```

```

        }
    }
    stage('Deploy to prod') {
        steps {
            sshagent(['ansible-key']) {
                sh '''
                    ssh -t -t -o StrictHostKeyChecking=no -o
                    ProxyCommand="ssh -W %h:%p -o StrictHostKeyChecking=no ec2-user@${
                    {BASTION_IP}}" ec2-user@${ANSIBLE_IP} "ansible-playbook -i
                    /etc/ansible/prod_hosts /etc/ansible/deployment.yml"
                    '''
                }
            }
        }
        stage('check prod website availability') {
            steps {
                sh "sleep 200"
                sh "curl -s -o /dev/null -w \"%{http_code}\"
https://www.edenboutique.space"
                script {
                    def response = sh(script: "curl -s -o /dev/null -w \"%
${http_code}\" https://www.edenboutique.space", returnStdout: true).trim()
                    if (response == "200") {
                        slackSend(color: 'good', message: "The prod petclinic
website is up and running with HTTP status code ${response}.",
tokenCredentialId: 'slack')
                    } else {
                        slackSend(color: 'danger', message: "The prod
petclinic wordpress website appears to be down with HTTP status code $
${response}.", tokenCredentialId: 'slack')
                    }
                }
            }
        }
    }
}

```

Dockerfile explanation

This docker file is executed when the docker build stage of the pipeline is triggered.
here it sets the base image, locates the .war file and copies it to the tomcat webapp directory, sets
the directory it would run the application, install dependencies and run the application.

```

FROM openjdk:8-jre-slim
FROM ubuntu
FROM tomcat
#copy war file on the container
COPY **/*.war /usr/local/tomcat/webapps
WORKDIR /usr/local/tomcat/webapps
RUN apt update -y && apt install curl -y
RUN curl -O https://download.newrelic.com/newrelic/java-agent/newrelic-
agent/current/newrelic-java.zip && \
    apt-get install unzip -y && \
    unzip newrelic-java.zip -d /usr/local/tomcat/webapps
WORKDIR /usr/local/tomcat/webapps
COPY newrelic.yml /usr/local/tomcat/webapps/newrelic/newrelic.yml

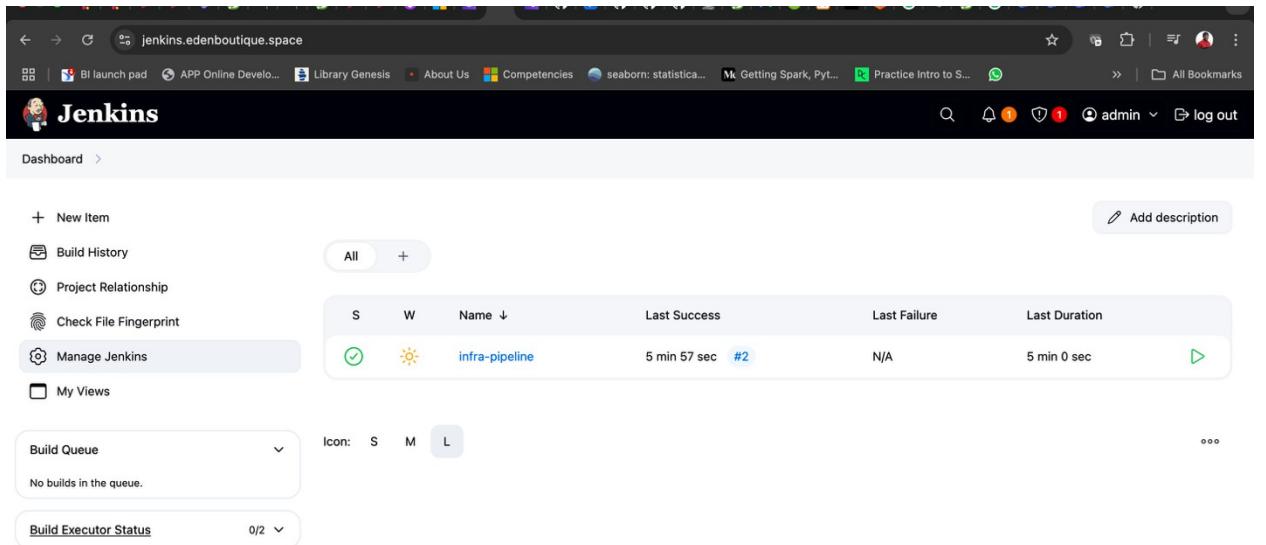
```

```
ENTRYPOINT [ "java", "-javaagent:/usr/local/tomcat/webapps/newrelic/newrelic.jar", "-jar", "spring-petclinic-2.4.2.war", "--server.port=8080
```

J. JENKINS PIPELINE SETUP FOR APPLICATION AND APPLICATION DEPLOYMENT

To setup the Jenkins pipeline, we need to install plugins,setup nexus,sonarqube and all other dependencies.

First go to manage Jenkins on your Jenkins url



The screenshot shows the Jenkins dashboard at jenkins.edenboutique.space. The dashboard features a sidebar with links like 'New Item', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins', and 'My Views'. The main area displays a table of pipelines. One pipeline is listed: 'infra-pipeline' (Status: Success, Last Success: 5 min 57 sec ago, Last Failure: N/A, Last Duration: 5 min 0 sec). Below the table are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (0/2 executors). The top navigation bar includes links for BI launch pad, APP Online Develo..., Library Genesis, About Us, Competencies, seaborn: statistica..., Getting Spark, PyT..., Practice Intro to S..., and a user account for 'admin'.

see resources deployed by Jenkins pipeline

INSTALL MAVEN PLUGIN

Next, click on plugins, here we want to install maven

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links for New Item, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins (which is selected), and My Views. Below the sidebar are sections for Build Queue (No builds in the queue) and Build Executor Status (0/2). The main area is titled "Manage Jenkins" and contains a message about Java 17 end-of-life. It also has a "System Configuration" section with links to System, Tools, Nodes, Clouds, Plugins, and Appearance. A search bar at the top right says "Search settings".

On the search bar, search for maven and select “Maven Integration” and click on install

The screenshot shows the Jenkins plugin manager page with a search bar containing "maven". Under the "Available plugins" tab, the "Maven Integration" plugin is selected for installation. The plugin details show it's version 3.26, released 1 month 5 days ago, and provides deep integration between Jenkins and Maven. Other available plugins listed include Config File Provider and Jira.

Install	Name	Released
<input checked="" type="checkbox"/>	Maven Integration 3.26 Build Tools	1 mo 5 days ago
<input type="checkbox"/>	Config File Provider 988.v0461fcc2b_9d1 Groovy-related External Site/Tool Integrations Maven	1 mo 2 days ago
<input type="checkbox"/>	Jira 3.16 External Site/Tool Integrations Maven jira	9 days 20 hr ago

Plugins

- Updates
- Available plugins
- Installed plugins
- Advanced settings
- Download progress

Download progress

Preparation	
Ionicons API	Success
Folders	Success
OWASP Markup Formatter	Success
ASM API	Success
JSON Path API	Success
Structs	Success
Pipeline: Step API	Success
Token Macro	Success
Build Timeout	Success
bouncycastle API	Success
Credentials	Success
Plain Credentials	Success
Variant	Success
SSH Credentials	Success

ADD FILE PATH FOR MAVEN AND JAVA

Next we need to add a file path for maven and java to Jenkins tool so as to use when building our pipeline. To get the file path where maven and java is intalled, ssh into your Jenkins server and type the command “mvn --version” you can see that item 2 is the maven file path awhile item 3 points to the java path. Copy them and add to the tool section of the Jenkins

```
using rhc at https://red.ht/registration
[ec2-user@jenkins ~]$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
154139e9cd8a477481c03511862f653d
[ec2-user@jenkins ~]$ mvn --version
Apache Maven 3.6.3 (Red Hat 3.6.3-22)
Maven home: /usr/share/maven 1
Java version: 17.0.15, vendor: Red Hat, Inc., runtime: /usr/lib/jvm/java-17-openjdk-17.0.15.0.6-2.el9.x86_64 2
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "5.14.0-427.68.1.el9_4.x86_64", arch: "amd64", family: "unix"
[ec2-user@jenkins ~]$ 3
```

Ln 1, Col 27 (26 selected) Spaces: 4 UTF-8

Go to the tool section

Manage Jenkins

- + New Item
- Build History
- Project Relationship
- Check File Fingerprint
- Manage Jenkins
- My Views

System Configuration

Java 17 end of life in Jenkins

You are running Jenkins on Java 17, support for which will end on or after Mar 31, 2026. Refer to [the documentation](#) for more details.

Tools

Configure tools, their locations and automatic installers.

Nodes

Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Clouds

Add, remove, and configure cloud instances to provision agents on-demand.

Plugins

Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

Appearance

Configure the look and feel of Jenkins.

Scroll down to the “Maven Home ” and add file path for maven as shown in the picture.

Maven installations

Add Maven

Maven

Name: maven

MAVEN_HOME: /usr/share/maven

Install automatically ?

Add Maven



Scroll further you would see where to add the java file path

JDK installations

Add JDK

JDK

Name: java

JAVA_HOME: /usr/lib/jvm/java-17-openjdk-17.0.15.0.6-2.el9.x86_64

Install automatically ?

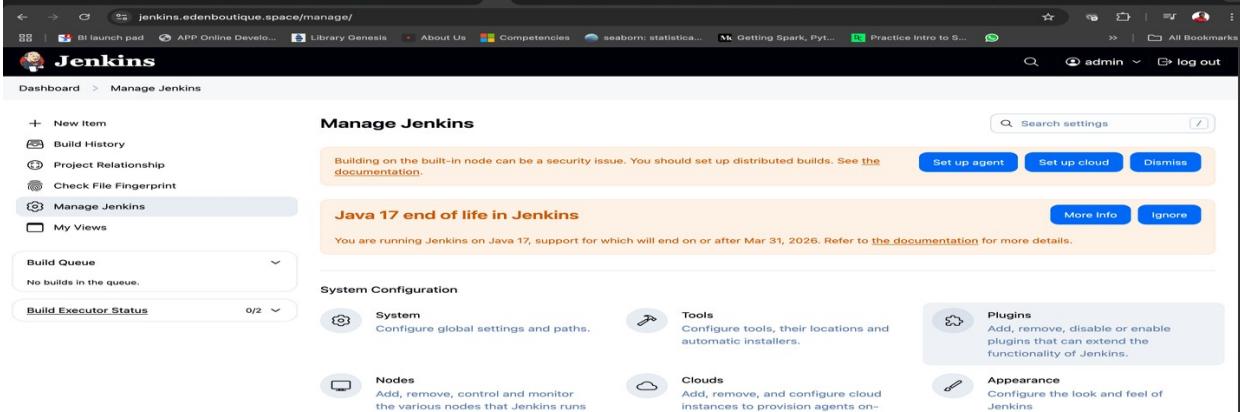
Add JDK



INSTALL SSH AGENT

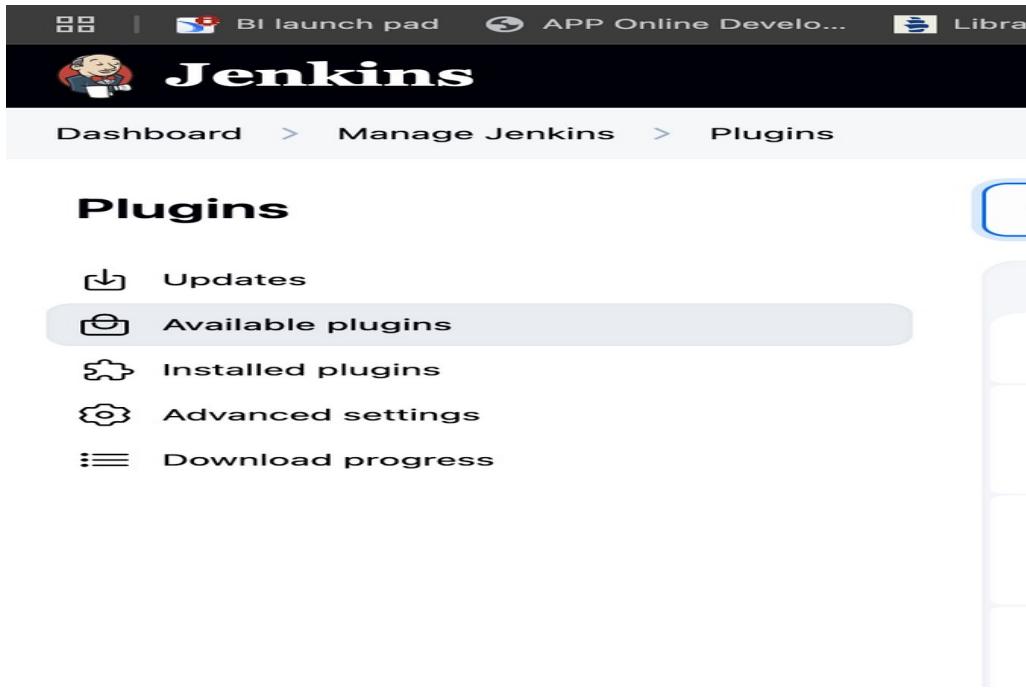
Here we need to install the SSHagent so as Jenkins can have access to ssh into other servers in the pipeline.

Go back to manage Jenkins and click on plugin.



The screenshot shows the Jenkins 'Manage Jenkins' interface. On the left, there's a sidebar with options like 'New Item', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins' (which is currently selected), and 'My Views'. The main content area has a heading 'Manage Jenkins' with a note about Java 17 end-of-life. It includes sections for 'System Configuration' (with 'System' and 'Nodes' sub-options), 'Tools' (with 'Tools' and 'Clouds' sub-options), and 'Plugins' (with 'Plugins' and 'Appearance' sub-options). At the top right, there's a search bar labeled 'Search settings' and three buttons: 'Set up agent', 'Set up cloud', and 'Dismiss'.

Go to available plugin



Search for SSH agent and select it

The screenshot shows the Jenkins interface with the title bar "Jenkins". Below it, the navigation path is "Dashboard > Manage Jenkins > Plugins". The search bar at the top right contains the text "ssh". The results table has columns for "Install", "Name", and "Released". There are three entries:

Install	Name	Released
<input type="checkbox"/>	SSH 158.ve2a_e90fb_7319 Build Wrappers	6 mo 13 days ago
<input checked="" type="checkbox"/>	SSH Agent 386.v36cc0c7582f0 This plugin allows you to provide SSH credentials to builds via a ssh-agent in Jenkins.	17 days ago
<input type="checkbox"/>	Publish Over SSH 387.vec3df0f668cd Artifact Uploaders Build Tools Send build artifacts over SSH	3 mo 23 days ago
<input type="checkbox"/>	SSH Pipeline Steps 2.0.79.v1d1b_5f76dda_8 pipeline Jenkins pipeline steps which provides SSH facilities such as command execution or file transfer for continuous delivery.	3 mo 8 days ago

ADD THE ANSIBLE KEY CREDENTIALS

Here we need to add the username with private key that Jenkins would use to ssh into other servers. To do that,
Go to credentials

The screenshot shows the Jenkins Manage Jenkins interface. At the top, there is a warning message: "Java 17 end of life in Jenkins" with a link to documentation. Below this, the "System Configuration" section is visible, containing links for System, Tools, Plugins, Nodes, Clouds, Appearance, Security, and Credentials. The "Credentials" link is highlighted with a grey background.

On the drop down on global and select “add credentials”

The screenshot shows the Jenkins Credentials management page. It displays a table of existing credentials and a "Stores scoped to Jenkins" section where a new credential can be added. The "Add credentials" button is highlighted with a grey background.

T	P	Store	Domain	ID	Name
		System	(global)	slack	slack
		System	(global)	git-cred	onyiafranklin/******** (git-cred)

On the kind, select “ssh username with private key”
add EC2-user as the user then go and locate the private key

New credentials

Kind

SSH Username with private key

Scope

Global (Jenkins, nodes, items, all child items, etc.)

ID

ansible-key

Description

ansible-key

Username

ec2-user

Treat username as secret

Private Key

Enter directly

Create

To locate the private key, ssh into your Jenkins server , cd into the Jenkins workspace cd /var/lib/Jenkins/workspace/(your pipeline name infra-pipeline) and list all file there you would see your pem key“auto-discov-key.pem”

```
OS name: "linux", version: "5.14.0-427.68.1.el9_4.x86_64", arch: "amd64", family: "unix"
[ec2-user@jenkins ~]$ cd /var/lib/jenkins/workspace/infra-pipeline
[ec2-user@jenkins infra-pipeline]$ ls -al
total 56
drwxr-xr-x. 7 root jenkins 4096 May 31 10:18 .
drwxr-xr-x. 4 root jenkins 54 May 31 10:14 ..
-r--r-----. 1 root jenkins 3243 May 31 10:18 auto-discov-key.pem
drwxr-xr-x. 2 root jenkins 29 May 31 10:18 checkov-results.txt
-rw-r--r--. 1 root jenkins 1533 May 31 10:14 create-remote-state.sh
-rw-r--r--. 1 root jenkins 2376 May 31 10:14 destroy-remote.sh
drwxr-xr-x. 8 root jenkins 162 May 31 10:17 .git
```

cat the key so that you can copy the key

```
[ec2-user@jenkins ~]$ cat auto-discov-key.pem
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAyL...4 Root Jenkins 65 May 31 10:18 terraform.lock.hcl
-rw-r--r--. 1 root jenkins 6604 May 31 10:18 .terraform.lock.hcl
-rw-r--r--. 1 root jenkins 300 May 31 10:18 variable.tf
drwxr-xr-x. 2 root jenkins 127 May 31 10:14 vault-jenkins
[ec2-user@jenkins infra-pipeline]$ sudo cat auto-discov-key.pem
```

Copy the RSA private key

```

auto-discovery-mayist
stage-bashscript.sh .../ansible
main.tf .../sonarqube
variable.tf .../vault-jenkins
variable.tf autodiscovery-proj
Jenkinsfile

-----BEGIN RSA PRIVATE KEY-----
MIIEKAIKKCgEAxvnxAXdezF97AneS0gy1R0Fg1qUmMftLXYmvXuJzUuOpT2w
T85P7sBTYpkv3AVNj905McqInT6k3TB0kbZxf6tKPr9jy8x+MpGzuU+lZEV
HmK9Wpn401p318g61bjWk+3a41k/TEda1cbnLcza4aLoe91v1+SWCALCjPB
YxVwTsTB87wuH19KLTAzq3i2AAKPNB1o5oMT4aA10A0GrLvnZnIGMe7te091w
JwR3PB4TzRrd2f9YtbpH/Wvsu/Lcvf6uHicmp/wkVJlarieF5gpHIsCejymAt
apk4uCoF/BGtKeX37n0dPEN-QNp4ewLx6qHlnI6BNVi34F6gbefPT91Uhzbml
5jSEKEfJfcovKk6KwJG9vPvCKPRW/f7J86v0mIIHAgkZJA+HIVmk+RYj1iy
rY70evdsTsiq103yIp2g0r0Uy0TsPr0pEmzPtduYnd8q70ZFLMN+jJnVGY
6fF/20d0wwv1ffwF54m688by41Nxnef8R4/ybs0ZTQqeZZ7Gjqaq4665jeAHXSw+
8swkn+WaqpdLvk5HnTH1A7xF5Stq56KG5D5W9TK1H70LIdo20AnmJ6uI4jdcd
sUwkn/6BW5tueyekr30W5qn8ZfHwxyoqCHUEWohax5yfbSw2rgtNmncAwEA
AQKCAgQAK-YJLl2wqvz71eRegTTbWesXq4dH11GERiNbPPR7TNSRXdI/gm9r
5E9/YJbOKWpJZ10d6MxzMsmtTR+TDXsQwvdLlmPcOH/S4aPxy9s3p7j2rlq3
PMWWAPRPYMuox62fT2vNM90jioYSNmijg8blbheniCzJiow0tDAlmXv7RZr0dtSk
Tj801z1TMMb17d03e2zLzPyW0HdC/HKutb1yPQo+20d2tU4z0HeX88/R9w/ZHTV6
3G71gkW1y91z9dC09P8q3xqjcPjCnEqPu37v1S94ks0tIKB0l1gxb5yNAQt
HsAvxnV7j1mDGtfJqyUVsvCSwFB0cxtUJg4ac17AcDmp/xJLWhCsPAUvNDDH8
XTE13Ku/wjXzvUwCTgrB1+b1Vs2jWcvmsB107hm+jjie0sBDjP51znjp2m1/e
n2Ingh3DoyFkVdH4tZqagtyXj5ngzbXjcb8P0dxlpkkJfer5THzTvwU8/koe
6VuKcdZ2Bhd0s/FTKcnw3d3giITQ+jhEn1+4UpYedLDPlqlaqRv50iw0l0vx
2/s0HeYh10aa4HNzv0A+j+g4EkB+vTaPve+k42zXy3m65dwR0KCA0EA84c16zb
6mlWeQ10L4YwvYLPxmDttdW1TD5bcCzYbh9IL6huBp+qoNf0KAshKob7stuY
xn0uHhz61FXM1iNdRuzxBvmyK6A5G0+o1KAcEEEM2qLw0I1UuH8EhNR6Sxldj0ly
pNa4jez2wUe8sAAgkjJxsPHCMfb1vdyvNm/PghF6ubiaibrD0eBdqfIuk2qaxAk
wvpaP6/68HJRzDCNyA2ngjm+v6Cs0zRpdpvaur0WTShztbFnby/vKnxTzfs/C
owLuZq+v7YB1LXbKhczy6uroY487d6YpWDAF0h4qYjK2n08ZnWXY8KA0d80cuV6
1omByBk35euEvKAQEAYoNYZkxHcfLH0qLmr/060q/7ChLyN7kpLSr8n1
atVP07SES+oVCMPGJYsuwPNXDRhka+g4Glo1yaavSTHZzznsaCxLqnuGAbmu
juB8A1M2Lk2Ajbw5igs516dd0+Cca0a05Hp97VCYQy0c/2htr0)+oEByHau79
u09uhj3imNamzaNAdkud5LMUw555q0fN1w60TV2LEd12udpoD0dpHuhrs1yZh
4K1g1X1tPX-Nb25Xl0sNx0P/h2pHmBaPjyGHYlUK1/8q05LRCa9Gv3CcRne
8gYVqmqSu8Ydn10mSE19b+r1Prm91oyFa/Eng5m7PQKCAQEAS1kpJahmBVkoCuUu
wvCP7L1m1My5scGzoZ0UN5+rbh27lo1hJYJ1A+hXdh+Fd7nS/UupCj/q0rVXmo
qGNnwInclLfh5idvsvKG54lbRPCsWejHPt0qsEvvUmHKstUb5Vj4APoB031h/SW/nP
H47jSmNK9S9rATG1dmh0s0f01Kh0hbn0EXCs1o1IxRMe2lby/tTrsdU6R0
w1t1e35LduH2k9R1Ef3j6yyzfcd4018v8t1u5r190FNf03Ldp/nK8+LY9eBt16+
dmw011gpxYvkk7B2r7KryHCKTPG71zhpg0A9ghoTcKjwhKvuTxGjdzA70Wxm0
saAYqwKCAQAhii0KyHr8VHUOjwFVpMSnFcKTe0p23d/8Hw95ISU6yjg3vLx/
S/06wcygBkP7/HRYsRAN0ctf0IoqurisSTA5/W95qr1j70LkfcdEJHHmHHEGx
we4u/15H7m15XW28Qn+z5d8GnQjs5LkaoYd1fUlxqfX1y4p+HnR77REotkvZ
3dIE2ntixNE8XAHD055z8KNdz2677SyvFmkPM2EN165NafqwEGCsvCg0G45Wp
rs5ZT+54gw/VsMwtnnuqm649WmL185BjN+Hx02YY4jw0uJUJHMW1VvyeRmbvB
0j/wd+ozerC0SpRs9WmQv3aYKmW86jAoIBAfLak+EucAb/Lsg3+h/BQriewbIn
no1kv/2F7ywhIn0+oVNaXN0KmnuXnT28tUDRrgL9Ynf0vpG64h5tFJNjVls
vITg6C7U91KdujejTlVjd/p/umx4e4iW6w808nkbzr+pfq5jm+9C1f3UtgvqTw
xxFx1764v6LDTicf2GwB-Mf1X3D0/su3uR3l840Qd1NsXB4EogbUxfC4e2/o
/Ip1N2VXLnb01Gh+AY2Xl0qKHDSesxtWdJ9X9AYaJ2/jxUiFTg0upIFWrUc3GS2U
LsVT3LqCqM4bXmdzy/21c6kMo7jkq30vg90r604bL+f0aNshPK90AQ=A=
-----END RSA PRIVATE KEY-----
[ec2-user@jenkins infra-pipeline]$ 

```

Paste the RSA key on the private key section

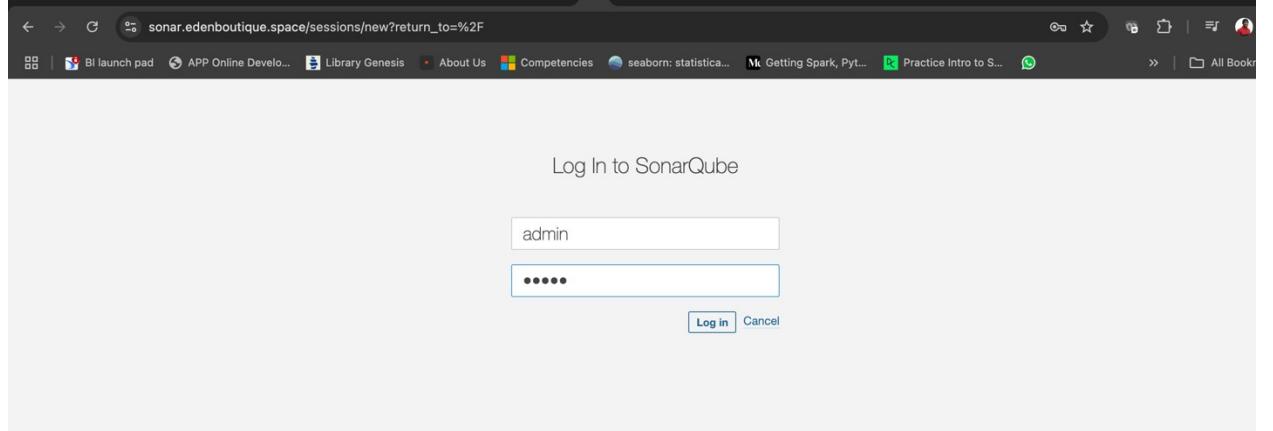
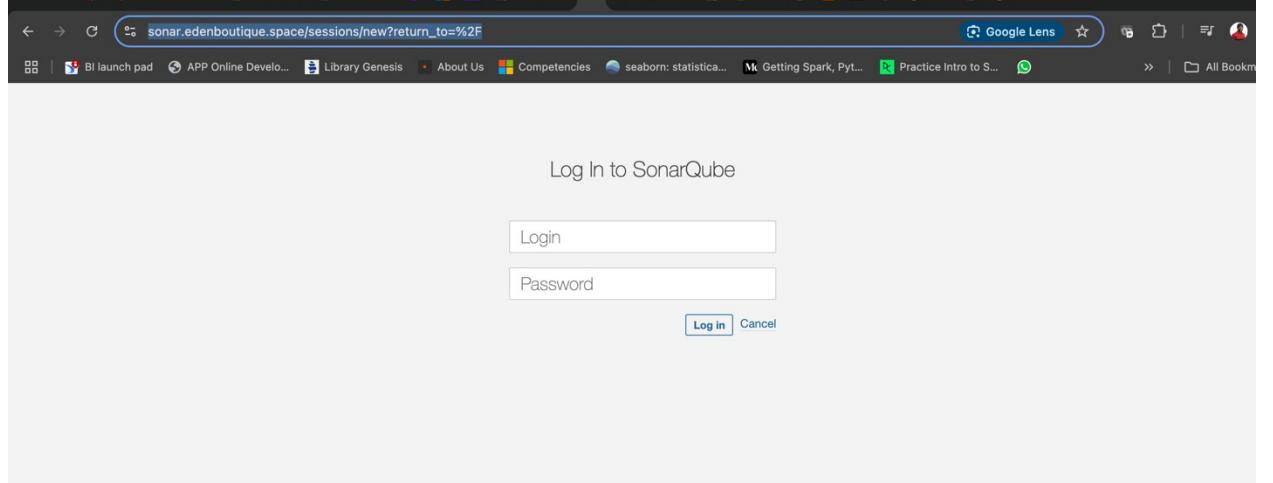
ID	ansible-key
Description	ansible-key
Username	ec2-user
<input type="checkbox"/> Treat username as secret	
Private Key	<input checked="" type="radio"/> Enter directly
Key	
<pre>/Ip1N2VXLnb01Gh+AY2Xl0qKHDSesxtWdJ9X9AYaJ2/jxUiFTg0upIFWrUc3GS2U LsVT3LqCqM4bXmdzy/21c6kMo7jkq30vg90r604bL+f0aNshPK90AQ=A= -----END RSA PRIVATE KEY-----</pre>	
Enter New Secret Below	
<input type="button" value="Create"/>	

CONFIGURE SONARQUBE

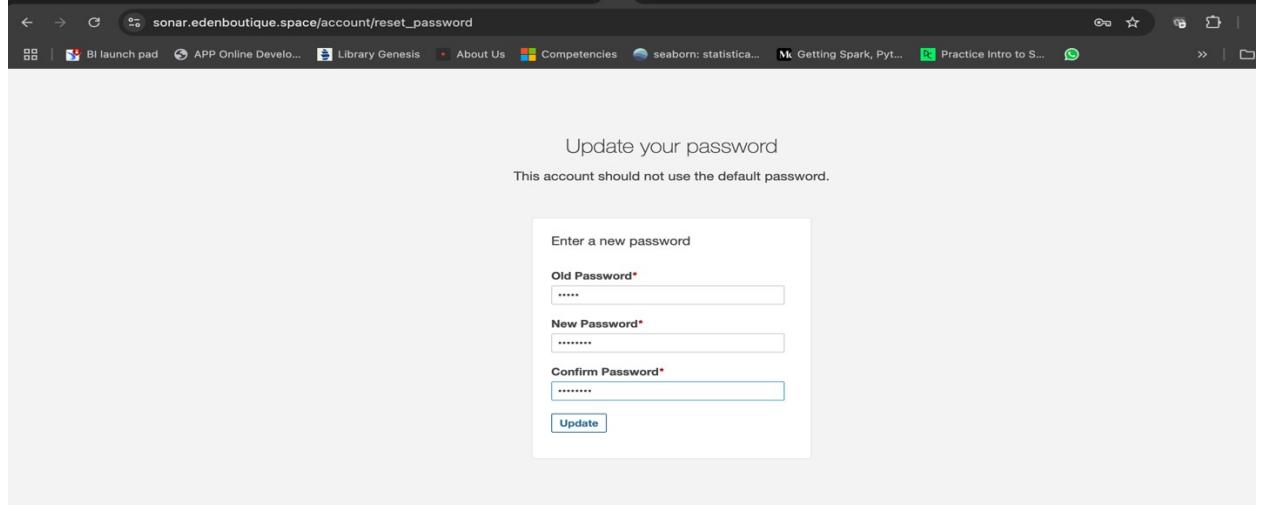
To configure sonarqube, we need to create a webhook so that sonarqube can connect to

Jenkins as well as create a webhook on Jenkins so that Jenkins would be aware of the sonarqube aswell

First visit the sonarqube url and login with default credentials username“admin” password ”admin”



You would be prompted to change the password, here you can change it to the credentials you set on the userdata script



Go to configuration and click on “webhooks” then create a webhook

The screenshot shows the SonarQube administration interface at <https://sonar.edenboutique.space/admin/settings>. The left sidebar is open, showing various configuration categories like General Settings, Encryption, Custom Metrics, and Webhooks. The main content area is titled 'Duplications' and contains sections for 'Cross project duplication detection' and 'Email'. In the 'Email' section, there is a field labeled 'Email prefix' with the value '[SONARQUBE]' and a note '(default)'. A 'Create' button is located at the bottom right of the main content area.

Click on create

The screenshot shows the SonarQube administration interface at <https://sonar.edenboutique.space/admin/webhooks>. The left sidebar is open, showing the 'Webhooks' category. The main content area is titled 'Webhooks' and contains a brief description: 'Webhooks are used to notify external services when a project analysis is done. An HTTP POST request including a JSON payload is sent to each of the provided URLs. Learn more in the [Webhooks documentation](#)'. Below this, a message states 'No webhook defined.' and features a 'Create' button at the top right.

Fill in the “Create webhook” with Name and the url of the Jenkins as shown below.
Click on create

The screenshot shows the SonarQube Administration interface. A modal window titled "Create Webhook" is open. It contains fields for "Name*" (set to "sonarqube"), "URL*" (set to "https://jenkins.edenboutique.space/sonarqube-webhook"), and "Secret" (an empty field). Below the URL field, there is a note about using a secret to generate an HMAC hex digest. At the bottom of the modal are "Create" and "Cancel" buttons. The background shows the SonarQube navigation bar with links like "Quality Profiles", "Quality Gates", and "Administration".

SonarQube™ technology is powered by SonarSource SA
Community Edition - Version 8.6 (build 39681) ~ LGPL v3 ~ Community ~ Documentation ~ Plugins ~ Web API ~ About

sonar.edenboutique.space/admin/webhooks

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Administration

Configuration ▾ Security ▾ Projects ▾ System Marketplace

Webhooks

Webhooks are used to notify external services when a project analysis is done. An HTTP POST request including a JSON payload is sent to each of the provided URLs. Learn more in the [Webhooks documentation](#).

Name	URL	Secret?	Last delivery
sonarqube	https://jenkins.edenboutique.space/sonarqube-webhook	No	Never

Next click on security and then users

The screenshot shows the SonarQube Administration interface. The top navigation bar includes links for BI launch pad, APP Online Develop..., Library Genesis, About Us, Competencies, and Administration. The main navigation bar has tabs for sonarqube, Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration, with Administration selected. Below this is a sub-navigation bar with Configuration, Security (selected), Projects, System, and Marketplace. A dropdown menu for Security is open, showing options: Users, Groups, Global Permissions, and Permission Templates. The main content area is titled "Administration" and "Webhooks". It explains that webhooks trigger services when a project analysis is done via an HTTP POST request. A table lists a single webhook entry:

Name	URL
sonarqube	https://jenkins.edenboutique.space/sonarqube-webhook

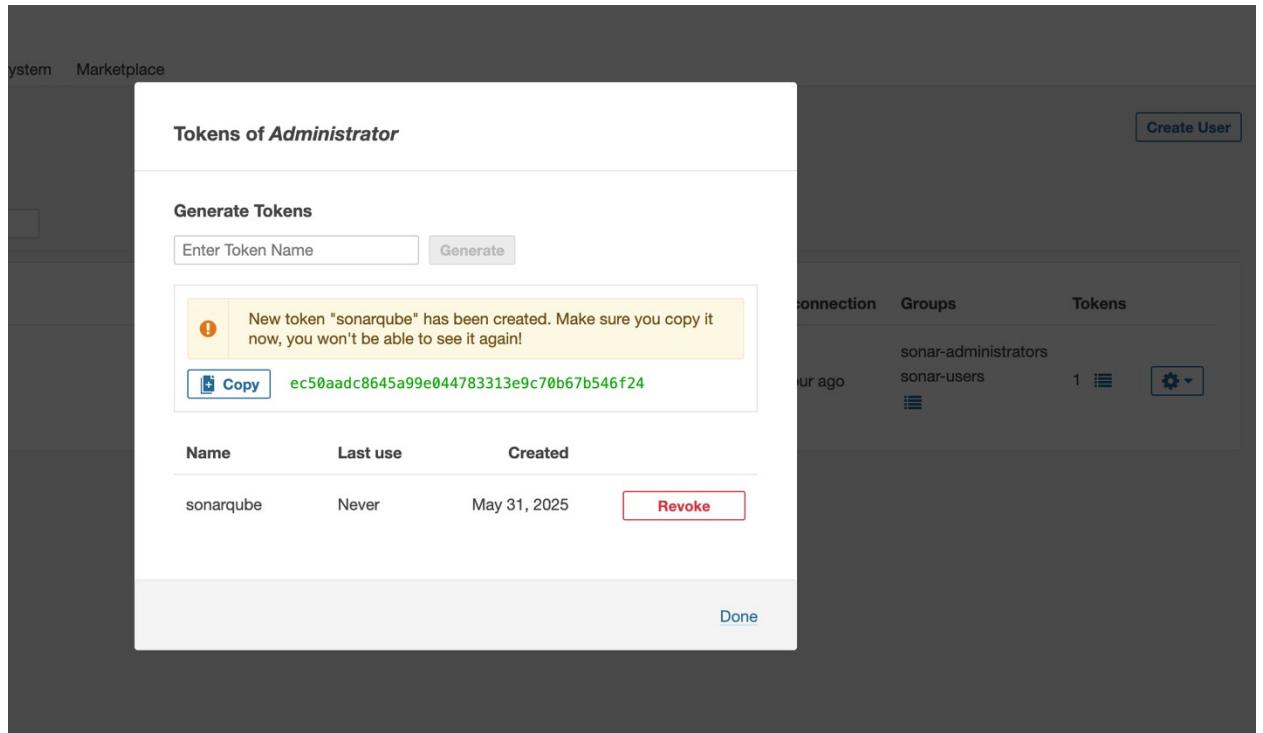
click on update tokens

The screenshot shows the SonarQube User Administration page. The URL is sonar.edenboutique.space/admin/users. The main navigation bar includes links for sonarqube, Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration, with Administration selected. A sub-navigation bar shows Configuration, Security (selected), Projects, System, and Marketplace. The main content area is titled "Administration" and "Users". It says "Create and administer individual users." and features a search bar "Search by login or name...". A table lists users:

	SCM Accounts	Last connection	Groups	Tokens
A Administrator admin		< 1 hour ago	sonar-administrators sonar-users	0

An "Update Tokens" button is located at the bottom right of the table row.

Enter token name an generate token, copy the created token



Go to plugins and install “sonar scanner plugins”

The screenshot shows the 'Manage Jenkins' page in Jenkins. On the left, there's a sidebar with links like 'Dashboard', 'Manage Jenkins' (which is selected), 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins', and 'My Views'. The main area has a heading 'manage Jenkins'. It features a warning about Java 17 end-of-life: 'Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.' with buttons for 'Set up agent', 'Set up cloud', and 'Dismiss'. Another warning about Java 17 end-of-life: 'Java 17 end of life in Jenkins' with buttons for 'More Info' and 'Ignore'. Below these are sections for 'System Configuration' (with 'System', 'Nodes', 'Tools', 'Clouds', and 'Plugins' tabs), 'Security', 'Credentials', and 'Credential Providers'. The 'Plugins' section specifically mentions adding, removing, disabling or enabling plugins to extend Jenkins functionality.

The screenshot shows the Jenkins plugin manager interface. On the left, there's a sidebar with options like 'Updates', 'Available plugins' (which is selected), 'Installed plugins', 'Advanced settings', and 'Download progress'. The main area has a search bar at the top with 'sonar' typed in. Below it is a table of results:

Install	Name	Released
<input checked="" type="checkbox"/>	SonarQube Scanner 2.18	4 mo 2 days ago
<input type="checkbox"/>	Sonar Quality Gates 352.vdcdb_d7994fb_6	3 mo 7 days ago
<input type="checkbox"/>	Quality Gates 2.5	9 yr 0 mo ago

A warning message is displayed for the 'Quality Gates' plugin: "Warning: This plugin version may not be safe to use. Please review the following security notices: • Credentials transmitted in plain text".

Next go to credentials, choose secret text and add that token you copied

The screenshot shows the Jenkins system configuration page. On the left, there's a sidebar with 'Manage Jenkins' selected. The main area has a heading 'Java 17 end of life in Jenkins' with 'More Info' and 'Ignore' buttons. Below it is the 'System Configuration' section:

- System**: Configure global settings and paths.
- Tools**: Configure tools, their locations and automatic installers.
- Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Nodes**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Clouds**: Add, remove, and configure cloud instances to provision agents on-demand.
- Appearance**: Configure the look and feel of Jenkins.

At the bottom, there's a 'Security' section with 'Credentials' highlighted, which is described as "Configure credentials".

The screenshot shows a web browser window for the Jenkins interface at the URL `jenkins.edenboutique.space/manage/credentials/store/system/domain/_/newCredentials`. The browser's address bar and various tabs are visible at the top. The main content area is titled "New credentials". It contains several input fields:

- Kind:** A dropdown menu set to "Secret text".
- Scope:** A dropdown menu set to "Global (Jenkins, nodes, items, all child items, etc)".
- Secret:** A text input field containing ".....".
- ID:** A text input field containing "sonarqube".
- Description:** A text input field containing "sonarqube".

A blue "Create" button is located at the bottom left of the form.

Go back to manage Jenkins and click on System

Manage Jenkins

Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).

Java 17 end of life in Jenkins

You are running Jenkins on Java 17, support for which will end on or after Mar 31, 2026. Refer to [the documentation](#).

System Configuration

- System**: Configure global settings and paths.
- Nodes**: Add, remove, control and monitor the various nodes that Jenkins runs.
- Tools**: Configure tools, their locations and automatic installers.
- Clouds**: Add, remove, and configure cloud instances to provision agents on-

Scroll down to sonarqube servers, enable the environment variable and then add the sonarqube url on the server url ,the select the credentials you created with the token.

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

Environment variables

SonarQube installations

List of SonarQube installations

Name	sonarqube
Server URL	Default is http://localhost:9000 <input type="text" value="https://sonar.edenboutique.space"/>
Server authentication token	SonarQube authentication token. Mandatory when anonymous access is disabled. <input type="text" value="sonarqube"/>
+ Add	
Advanced Save Apply	

SET UP NEXUS SERVER

To setup our nexus server, we need to visit the url and creat two registries one which is

maven hosted to store the artefact generated from maven and the other which is docker hosted to store our docker image.

Go to the nexus url and click on sign in.

Remember, we setup nexus server security group in a way that we allow ssh from the bastion.

so lets pick up our bastion host ip address

copy the ip address

eu-west-2.console.aws.amazon.com/ec2/home?region=eu-west-2#InstanceDetails:instanceId=i-086d3a0cd1e09b20e

EC2

- Instances
- Image
- Elastic Block Store

Instance summary for i-086d3a0cd1e09b20e (auto-discov-bastion-asg) [Info](#)

Updated less than a minute ago

Instance ID	i-086d3a0cd1e09b20e	Public IPv4 address	18.130.51.153 open address	Private IPv4 addresses	10.0.1.198
IPv6 address	-	Instance state	Running	Public DNS	-
Hostname type	IP name: ip-10-0-1-198.eu-west-2.compute.internal	Private IP DNS name (IPv4 only)	ip-10-0-1-198.eu-west-2.compute.internal	Elastic IP addresses	-
Answer private resource DNS name	-	Instance type	t2.medium	AWS Compute Optimizer finding	Opt-in to AWS Compute Optimizer for recommendations.
Auto-assigned IP address	18.130.51.153 [Public IP]	VPC ID	vpc-036c4cf246e65b96a (auto-discov-vpc) [?]	Auto Scaling Group name	auto-discov-bastion-asg
IAM Role	-	Subnet ID	subnet-0031203d6b59149ae (auto-discov-pub_sub1) [?]	Instance ARN	-

```
ssh into the bastion host
-----END RSA PRIVATE KEY-----
[ec2-user@jenkins infra-pipeline]$ ssh -i auto-discov-key.pem ec2-user@18.130.51.153
The authenticity of host '18.130.51.153 (18.130.51.153)' can't be established.
ED25519 key fingerprint is SHA256:UhT8ECE0i6jDIO/Bj0tXQ1Je7BL2zJK6cTZgtPHCgM.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '18.130.51.153' (ED25519) to the list of known hosts.
Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
[ec2-user@bastion ~]$ 
```

Ln 1, Col 27 (26 selected) Spaces: 4

Go back to the console and pick the ip address of the nexus

eu-west-2.console.aws.amazon.com/ec2/home?region=eu-west-2#InstanceDetails:instanceId=i-0b7bcc15243c47581

EC2

- Instances
- Image
- Elastic Block Store

Instance summary for i-0b7bcc15243c47581 (auto-discov-nexus-server) [Info](#)

Updated less than a minute ago

Instance ID	i-0b7bcc15243c47581	Public IPv4 address	3.8.139.214 open address	Private IPv4 addresses	10.0.1.156
IPv6 address	-	Instance state	Running	Public DNS	-
Hostname type	IP name: ip-10-0-1-156.eu-west-2.compute.internal	Private IP DNS name (IPv4 only)	ip-10-0-1-156.eu-west-2.compute.internal	Elastic IP addresses	-
Answer private resource DNS name	-	Instance type	t2.medium	AWS Compute Optimizer finding	Opt-in to AWS Compute Optimizer for recommendations.
Auto-assigned IP address	3.8.139.214 [Public IP]	VPC ID	vpc-036c4cf246e65b96a (auto-discov-vpc) [?]	Auto Scaling Group name	-
IAM Role	-	Subnet ID	subnet-0031203d6b59149ae (auto-discov-pub_sub1) [?]	Instance ARN	-

Private IPv4 address copied

ssh into the nexus server

```
Create an account or view all your systems at https://red.ht/insights-dashboard
[ec2-user@bastion ~]$ ssh 10.0.1.156
The authenticity of host '10.0.1.156 (10.0.1.156)' can't be established.
ED25519 key fingerprint is SHA256:k/ZPoTKBm5aGs9zyrxSgjgw4qrQVJS20ID3Z71a0IfE.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.1.156' (ED25519) to the list of known hosts.
Register this system with Red Hat Insights: rhc connect

Example:
# rhc connect --activation-key <key> --organization <org>

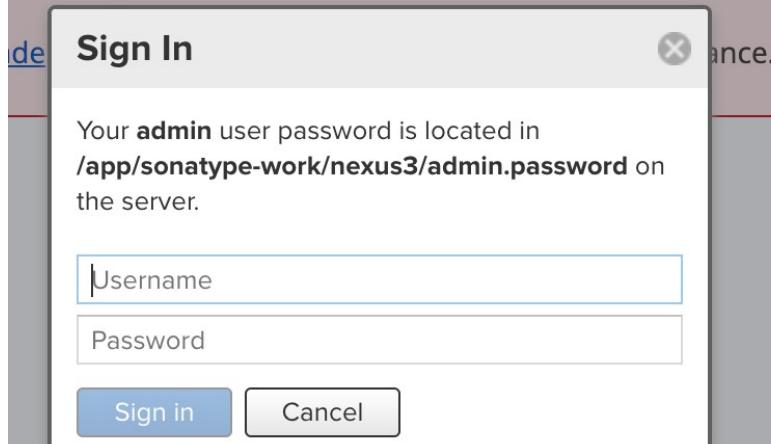
The rhc client and Red Hat Insights will enable analytics and additional
management capabilities on your system.
View your connected systems at https://console.redhat.com/insights

You can learn more about how to register your system
using rhc at https://red.ht/registration
[ec2-user@Nexus ~]$
```

Ln 1, Col 27 (26 selected) Spaces: 4

sudo cat into the nexus file path /app/sonatype-work/nexus3/admin.password to get the password credential

repository is vulnerable to a critical remote code execution security vulnerability.



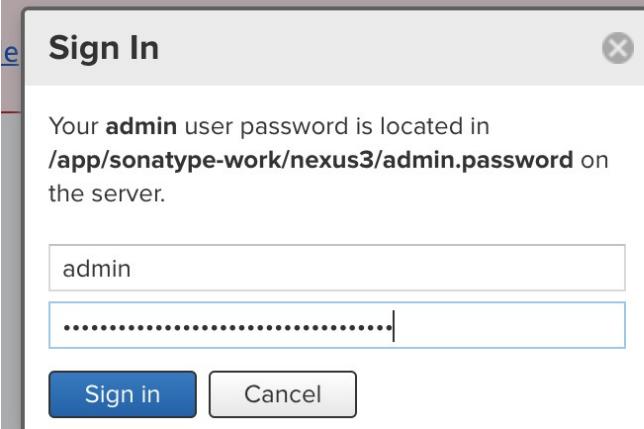
Copy the password credentials

[View your connected systems at https://console.redhat.com/insights](https://console.redhat.com/insights)

```
You can learn more about how to register your system
using rhc at https://red.ht/registration
[ec2-user@Nexus ~]$ sudo cat /app/sonatype-work/nexus3/admin.password
0a300d52-1efa-41ce-af7d-74ca3d65638d[ec2-user@Nexus ~]$
```

Sign in using the admin and password copied

repository is vulnerable to a critical remote code execution security vulnerability.



Click on next

We recommend [immediate upgrade](#) to 3.25.1 or later, especially if this is a public instance.

Setup

1 of 4

This wizard will help you complete required setup tasks.

[Next](#)

you would be asked to change the password

We recommend [immediate upgrade](#) to 3.25.1 or later, especially if this is a public instance.

Please choose a password for the admin user

2 of 4

New password:

.....

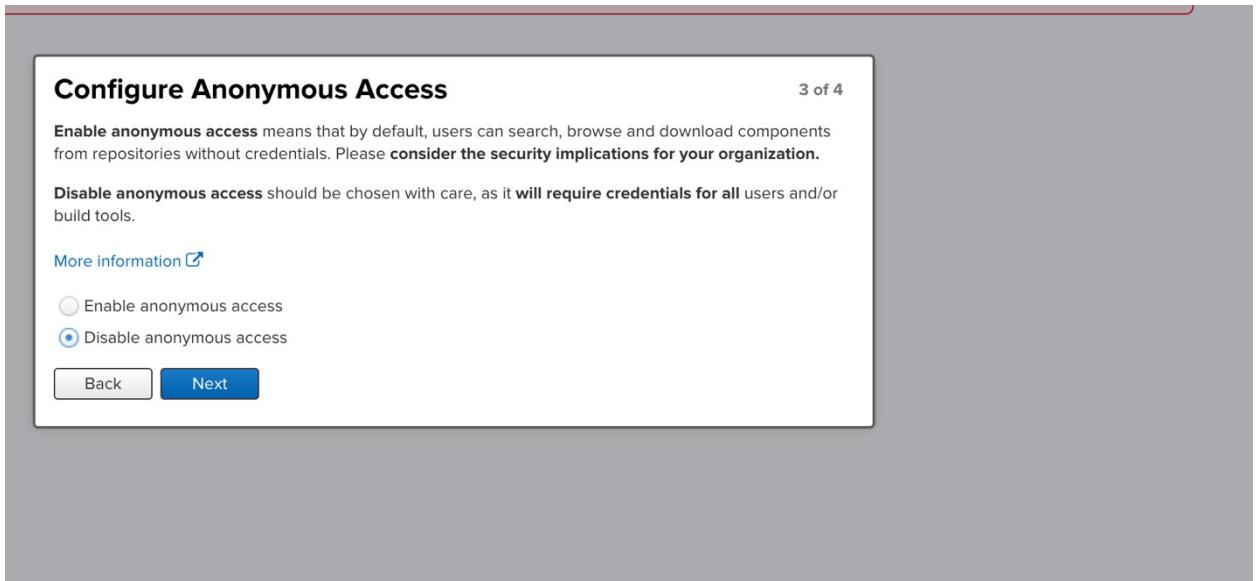
Confirm password:

.....

[Back](#)

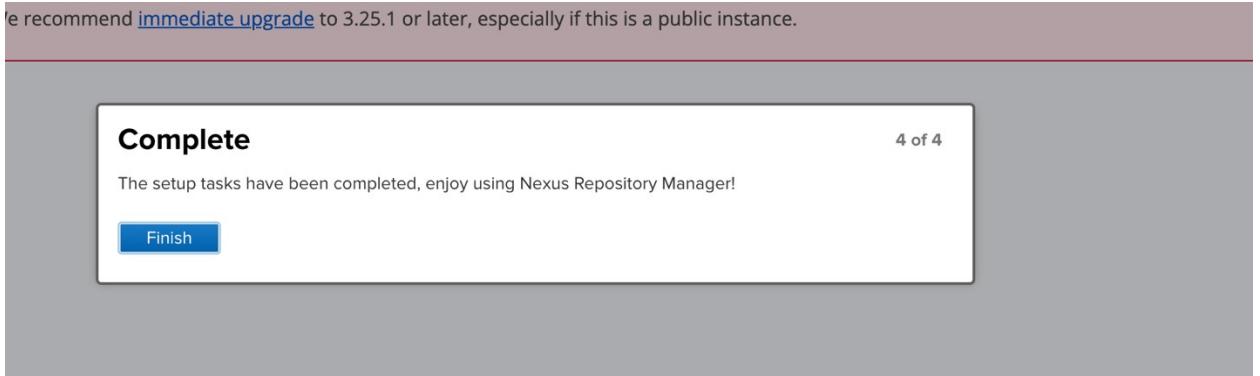
[Next](#)

Disable anonymous access



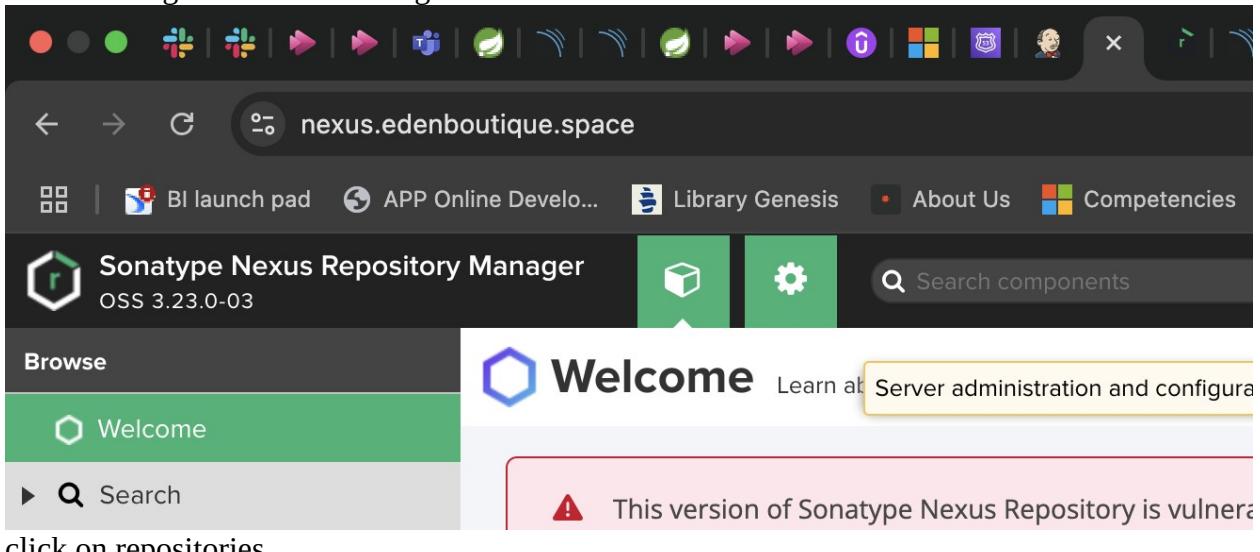
Click on finish

We recommend [immediate upgrade](#) to 3.25.1 or later, especially if this is a public instance.



Now is time to create the repository, we would be creating the maven and docker hosted repository.

click on the green colored settings icon



click on repositories

The screenshot shows the Sonatype Nexus Repository Manager interface. The URL is `nexus.edenboutique.space/#admin/repository`. The top navigation bar includes links for BI launch pad, APP Online Develo..., Library Genesis, About Us, Competencies, seaborn: statistica..., Getting Spark, Pyt..., Practice Intro to S..., and All Bookmarks. The user is signed in as 'admin'. The main header is 'Sonatype Nexus Repository Manager OSS 3.23.0-03' with a gear icon for settings. The left sidebar under 'Administration' has a 'Repository' section expanded, showing 'Repositories', 'Blob Stores', 'Content Selectors', 'Cleanup Policies', 'Routing Rules', and a collapsed 'Security' section with 'Privileges', 'Roles', and 'Users'. Below this is an 'Anonymous Access' section. The right panel title is 'Repository' with a subtitle 'Repository administration'. It contains tabs for 'Blob Stores', 'Cleanup Policies', 'Repositories' (which is selected), 'Routing Rules', and 'Content Selectors'. A search bar at the top of the right panel says 'Search components'.

Click on create repository

The screenshot shows the 'Repositories' management page. The URL is `nexus.edenboutique.space/#admin/repository/repositories`. The top navigation bar and user information are identical to the previous screenshot. The main header is 'Sonatype Nexus Repository Manager OSS 3.23.0-03'. The left sidebar is identical to the previous screenshot. The right panel title is 'Repositories' with a subtitle 'Manage repositories'. It features a 'Create repository' button and a table listing existing repositories:

Name ↑	Type	Format	Status
maven-central	proxy	maven2	Online - Ready to Connect
maven-public	group	maven2	Online
maven-releases	hosted	maven2	Online
maven-snapshots	hosted	maven2	Online
nuget-group	group	nuget	Online
nuget-hosted	hosted	nuget	Online
nuget.org-proxy	proxy	nuget	Online - Ready to Connect

Search for maven2 hosted repository

The screenshot shows the Sonatype Nexus Repository Manager interface. The left sidebar is titled 'Administration' and contains several sections: 'Repository' (selected), 'Blob Stores', 'Content Selectors', 'Cleanup Policies', 'Routing Rules', 'Security' (with 'Privileges', 'Roles', 'Users', 'Anonymous Access', 'LDAP', 'Realms', and 'SSL Certificates'), 'IQ Server', 'Support', and 'System'. The main content area is titled 'Repositories' and shows a list of 'Recipes' under 'Select Recipe'. The list includes: apt (hosted), apt (proxy), bower (group), bower (hosted), bower (proxy), cocoapods (proxy), conan (proxy), conda (proxy), docker (group), docker (hosted), docker (proxy), grifl (hosted), go (group), go (proxy), helm (hosted), helm (proxy), maven2 (group), maven2 (hosted), maven2 (proxy), npm (group), npm (hosted), npm (proxy), nuget (group), and nuget (hosted). A search bar at the top right says 'Search components'.

Give your repository a name.

On version policy, choose “mixed”.

On deployment policy, choose “allow redeploy”.

The screenshot shows the 'Create Repository' configuration page. The left sidebar is identical to the previous screenshot. The main content area is titled 'Select Recipe' and shows 'Create Repository: maven2 (hosted)'. The configuration fields are as follows:

- Name:** nexus-repo
- Online:** If checked, the repository accepts incoming requests
- Maven 2**
 - Version policy:** Mixed
 - Layout policy:** Strict
- Storage**
 - Blob store:** default
 - Strict Content Type Validation:** Validate that all content uploaded to this repository is of a MIME type appropriate for the repository format
- Hosted**
 - Deployment policy:** Allow redeploy
- Cleanup**

then click on create repository

view the repository created

Name	Type	Format	Status	URL	Health check	IQ Policy Viola...
maven-central	proxy	maven2	Online - Ready to Connect			
maven-public	group	maven2	Online			
maven-releases	hosted	maven2	Online			
maven-snapshots	hosted	maven2	Online			
nexus-repo	hosted	maven2	Online			
nuget-group	group	nuget	Online			
nuget-hosted	hosted	nuget	Online			
nuget.org-proxy	proxy	nuget	Online - Ready to Connect			

Next, search for docker hosted repository

The screenshot shows the Sonatype Nexus Repository Manager interface. The left sidebar is titled 'Administration' and includes sections for 'Repository' (Repositories, Blob Stores, Content Selectors, Cleanup Policies, Routing Rules), 'Security' (Privileges, Roles, Users, Anonymous Access), and 'System' (IQ Server, Support, System). The main content area is titled 'Repositories' and shows a list of available recipes:

- apt (hosted)
- apt (proxy)
- bower (group)
- bower (hosted)
- bower (proxy)
- cocoapods (proxy)
- conan (proxy)
- conda (proxy)
- docker (group)
- docker (hosted)
- docker (proxy)
- gitlfs (hosted)
- go (group)

Give it a name.

Create an HTTP connector on port 8085 this is where the port where the docker image would be exposed

then enable docker vi API

The screenshot shows the 'Create Repository: docker (hosted)' configuration page. The left sidebar is identical to the previous screenshot. The main form has the following fields:

- Name:** docker-repo
- Online:** If checked, the repository accepts incoming requests

Repository Connectors

Connectors allow Docker clients to connect directly to hosted registries, but are not always required. Consult our documentation for which connector is appropriate for your use case. For information on scaling the repositories see our scaling documentation.

HTTP:
Create an HTTP connector at specified port. Normally used if the server is behind a secure proxy.
 8085

HTTPS:
Create an HTTPS connector at specified port. Normally used if the server is configured for https.

Allow anonymous docker pull:
 Allow anonymous docker pull (Docker Bearer Token Realm required)

Docker Registry API Support

Enable Docker V1 API:
 Allow clients to use the V1 API to interact with this repository

Storage

Blob store:
Blob store used to store repository contents
 default

Allow redeploy on deployment policy

The screenshot shows the Sonatype Nexus Repository Manager interface. The left sidebar is titled 'Administration' and includes sections for Repository (Repositories, Blob Stores, Content Selectors, Cleanup Policies, Routing Rules), Security (Privileges, Roles, Users, Anonymous Access, LDAP, Realms, SSL Certificates), Support, and System. The main content area is titled 'Repositories / Select Recipe / Create Repository: docker (hosted)'. It contains several configuration sections: 'Docker Registry API Support' (Enable Docker V1 API: checked), 'Storage' (Blob store: default), 'Strict Content Type Validation' (checked), 'Hosted' (Deployment policy: Allow redeploy), and 'Cleanup' (Cleanup Policies: Available and Applied lists). A note at the bottom says 'then click on create repository'.

The screenshot shows the same configuration page as the previous one, but with the 'Create repository' button at the bottom highlighted in blue. The rest of the interface is identical to the first screenshot.

view the docker repo created

Name	Type	Format	Status	URL	Health check	IQ Policy Violations
docker-repo	hosted	docker	Online			
maven-central	proxy	maven2	Online - Ready to Connect			
maven-public	group	maven2	Online			
maven-releases	hosted	maven2	Online			
maven-snapshots	hosted	maven2	Online			
nexus-repo	hosted	maven2	Online			
nuget-group	group	nuget	Online			
nuget-hosted	hosted	nuget	Online			
nuget.org-proxy	proxy	nuget	Online - Ready to Connect			

Go to security an click on realms

click on "docker bearer token realms"

The screenshot shows the 'Realms' management interface in the Sonatype Nexus Repository Manager. On the left, a sidebar menu is visible with sections like 'Repository', 'Security', and 'Realms' (which is currently selected and highlighted in green). The main content area is titled 'Realms: Manage the active security realms and their order'. It displays two lists: 'Available' on the left and 'Active' on the right. In the 'Available' list, several realms are listed: Conan Bearer Token Realm, Default Role Realm, Docker Bearer Token Realm, LDAP Realm, npm Bearer Token Realm, NuGet API-Key Realm, and Rut Auth Realm. In the 'Active' list, 'Local Authenticating Realm' and 'Local Authorizing Realm' are present. Between these lists are four control buttons: up and down arrows for reordering, and 'Save' and 'Discard' buttons at the bottom.

Click on the docker bearer token realms and add to the active side

This screenshot shows the same 'Realms' management interface as the previous one, but with a specific action taken. The 'Docker Bearer Token Realm' has been selected in the 'Available' list (indicated by a gray background) and is being moved to the 'Active' list. A yellow box highlights the 'Add to Selected' button, which is positioned between the two lists. The other realms in the 'Available' list remain unselected.

Now install nexus artefact uploader.
Go to “manage jenkins”
click on systems

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links for New Item, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins (which is selected), and My Views. Below this is a dropdown for Build Queue and a Build Executor Status section showing 0/2. The main area is titled "Manage Jenkins" and contains several configuration sections:

- System Configuration:** Includes links for System (Configure global settings and paths), Tools (Configure tools, their locations and automatic installers), Nodes (Add, remove, control and monitor the various nodes that Jenkins runs jobs on), Clouds (Add, remove, and configure cloud instances to provision agents on-demand), Plugins (Add, remove, disable or enable plugins that can extend the functionality of Jenkins), and Appearance (Configure the look and feel of Jenkins).
- A warning message: "Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation." with buttons for "Set up agent", "Set up cloud", and "Dismiss".
- A notice about Java 17 end of life: "Java 17 end of life in Jenkins" with buttons for "More Info" and "Ignore".

The screenshot shows the Jenkins Plugin Manager page with a search bar containing "nexus". The results table lists two plugins:

Install	Name	Released
<input checked="" type="checkbox"/>	Nexus Artifact Uploader 2.14 Artifact Uploaders	This plugin to upload the artifact to Nexus Repository. This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information. 2 yr 6 mo ago
<input type="checkbox"/>	Maven Artifact ChoiceListProvider (Nexus) 1.17 Maven Build Parameters	This Plugin adds a new ChoiceListProvider for the "Extensible Choice Plugin" which is able to read Artifact information from a Nexus repository. 1 yr 7 mo ago

Now lets add the nexus credentials where artefact uploader would upload the artefact from maven.

GO TO MANAGE JENKINS
Click on credentials

The screenshot shows the Jenkins Manage Jenkins page. At the top, there is a warning message: "Java 17 end of life in Jenkins" with a note: "You are running Jenkins on Java 17, support for which will end on or after Mar 31, 2026. Refer to [the documentation](#) for more details." Below this, the "System Configuration" section is visible, featuring links for System, Tools, Plugins, Nodes, Clouds, Appearance, Security, Credentials, and User management.

System Configuration

- System**: Configure global settings and paths.
- Tools**: Configure tools, their locations and automatic installers.
- Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Nodes**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Clouds**: Add, remove, and configure cloud instances to provision agents on-demand.
- Appearance**: Configure the look and feel of Jenkins.
- Security**: Secure Jenkins; define who is allowed to access/use the system.
- Credentials**: Configure credentials (highlighted).
- Credential Providers**: Configure the credential providers and types.
- Users**: Create/delete/modify users that can log in to this Jenkins.

The URL in the browser is <https://jenkins.edenboutique.space/manage/credentials>.

Credentials

T	P	Store ↓	Domain	ID	Name
File	User	System	(global)	slack	slack
File	User	System	(global)	git-cred	onyiafranklin***** (git-cred)
Fingerprint	User	System	(global)	ansible-key	ec2-user (ansible-key)
File	User	System	(global)	sonarqube	sonarqube

Stores scoped to Jenkins

P	Store ↓	Domains
User	System	(global)

Icon: S M L

REST API Linking? 504?

Click on global to add credentials.
choose username and password as kind of credentials and add the credentials

The screenshot shows the Jenkins 'New credentials' creation interface. The 'Kind' dropdown is set to 'Username with password'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc.)'. The 'Username' field contains 'admin'. The 'Password' field contains '.....'. The 'ID' field is 'nexus-cred'. The 'Description' field is 'nexus-cred'. A blue 'Create' button is at the bottom.

Next we need to setup the credentials that would be used to pull the docker image from the nexus repository opened on port 8085
 to set it up we need to set up the username , password and (ip with port 8085)
 lets create a secret text credentials for nexus-username (admin)

The screenshot shows the Jenkins 'New credentials' creation interface. The 'Kind' dropdown is set to 'Secret text'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc.)'. The 'Secret' field contains '.....'. The 'ID' field is 'nexus-username'. The 'Description' field is 'nexus-username'. A blue 'Create' button is at the bottom.

create secret text nexus-password credentials (admin123)

New credentials

Kind
Secret text

Scope
Global (Jenkins, nodes, items, all child items, etc.)

Secret
.....

ID ?
nexus-password

Description ?
nexus-password

Create

Now to setup the ip with port, go to the aws console and grab the ip address of the nexus

EC2

Instance summary for i-0b7bcc15243c47581 (auto-discov-nexus-server) [Info](#)

Updated less than a minute ago

Attribute	Value
Instance ID	i-0b7bcc15243c47581
IPv6 address	-
Hostname type	IP name: ip-10-0-1-156.eu-west-2.compute.internal
Answer private resource DNS name	-
Auto-assigned IP address	3.8.139.214 [Public IP]
IAM Role	-
IMDSv2	Optional <small>EC2 recommends setting IMDSv2 to required Learn more</small>

Public IPv4 address [3.8.139.214 | open address](#)

Instance state [Running](#)

Private IP DNS name (IPv4 only) [ip-10-0-1-156.eu-west-2.compute.internal](#)

Instance type t2.medium

VPC ID [vpc-036c4cf246e65b96a \(auto-discov-vpc\)](#)

Subnet ID [subnet-0031203d6b59149ae \(auto-discov-pub_sub1\)](#)

Instance ARN [arn:aws:ec2:eu-west-2:896055377516:instance/i-0b7bcc15243c47581](#)

Private IPv4 addresses [10.0.1.156](#)

Public DNS -

Elastic IP addresses -

AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. | Learn more

Auto Scaling Group name -

Managed false

create a secret text credential with the ip and port of the nexus docker repo in the format (ipaddress:8085)

The screenshot shows the Jenkins 'New credentials' creation interface. The 'Kind' dropdown is set to 'Secret text'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Secret' field contains '.....'. The 'ID' field is 'nexus-docker-repo'. The 'Description' field is 'nexus-docker-repo 3.8.139.214:8085'. A blue 'Create' button is at the bottom. The status message 'Credentials created successfully!' is displayed above the 'Create' button.

New credentials

Kind

Secret text

Scope ?
Global (Jenkins, nodes, items, all child items, etc)

Secret
.....

ID ?
nexus-docker-repo

Description ?
nexus-docker-repo 3.8.139.214:8085

Create

REST API Jenkins 2.504.2

The screenshot shows the Jenkins 'New credentials' creation interface. The 'Kind' dropdown is set to 'Secret text'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Secret' field contains '.....'. The 'ID' field is 'nexus-docker-repo'. The 'Description' field is 'nexus-docker-repo'. A red error message 'There was an error creating the credential' is displayed above the 'Create' button.

New credentials

Kind

Secret text

Scope ?
Global (Jenkins, nodes, items, all child items, etc)

Secret
.....

ID ?
nexus-docker-repo

Description ?
nexus-docker-repo

There was an error creating the credential

Create

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Next we need to create credential for the bastion ip so that Jenkins would use the credentials to logging to stage and prod server through tunnelling

copy the bastion host public ip from the console

Instance summary for i-086d3a0cd1e09b20e (auto-discov-bastion-asg)

- Instance ID:** i-086d3a0cd1e09b20e
- IPv6 address:** -
- Hostname type:** IP name: ip-10-0-1-198.eu-west-2.compute.internal
- Answer private resource DNS name:** -
- Auto-assigned IP address:** 18.130.51.153 [Public IP]
- IAM Role:** -
- IMDSv2:** Optional EC2 recommends setting IMDSv2 to required | Learn more
- Private IP DNS name (IPv4 only):** ip-10-0-1-198.eu-west-2.compute.internal
- Private IPv4 address:** 18.130.51.153 | open address
- Instance state:** Running
- Instance type:** t2.medium
- VPC ID:** vpc-036c4cf246e65b96a (auto-discov-vpc)
- Subnet ID:** subnet-0031203d6b59149ae (auto-discov-pub_sub1)
- Instance ARN:** arn:aws:ec2:eu-west-2:896035377516:instance/i-086d3a0cd1e09b20e
- Managed:** false
- Private IPv4 addresses:** 10.0.1.198
- Public DNS:** -
- Elastic IP addresses:** -
- AWS Compute Optimizer finding:** Opt-in to AWS Compute Optimizer for recommendations. | Learn more
- Auto Scaling Group name:** auto-discov-bastion-asg

New credentials

Kind: Secret text

Scope: Global (Jenkins, nodes, items, all child items, etc)

Secret:

ID: bastion-ip

Description: bastion-ip

Create

Next we need to create credential for the ansible ip so that Jenkins would use the credentials to logging to stage and prod server through tunnelling

copy the ansible host private ip from the console

Screenshot of the AWS EC2 Instance Details page for instance i-00f9cd756d34c84de.

Instance summary for i-00f9cd756d34c84de (auto-discov-ansible-server)

Instance ID	i-00f9cd756d34c84de	Public IPv4 address	Private IPv4 addresses
IPv6 address	-	Instance state	Public DNS
Hostname type	IP name: ip-10-0-3-201.eu-west-2.compute.internal	Private IP DNS name (IPv4 only)	Elastic IP addresses
Answer private resource DNS name	-	Instance type	AWS Compute Optimizer finding
Auto-assigned IP address	-	VPC ID	Learn more
IAM Role	auto-discov-ansible-bucket-role	Subnet ID	Auto Scaling Group name
IMDSv2	Optional ⚠️ EC2 recommends setting IMDSv2 to required Learn more	Instance ARN	Managed

Create a new secret text credential with the ansible ip

Jenkins Credentials Management Page - New credentials

Kind: Secret text

Scope: Global (Jenkins, nodes, items, all child items, etc)

Secret:

ID: ansible-ip

Description: ansible-ip

Create

REST API Jenkins 2.504.2

Now we need to add a plugin for OWASP dependency scan and also create nvd key that it would use for faster scanning.

Go to manage jenkins and click on credentials

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like 'New Item', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins' (which is selected), and 'My Views'. Below the sidebar are two sections: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (0/2). The main content area has a heading 'Manage Jenkins'. It includes a warning about building on the built-in node being a security issue, with buttons for 'Set up agent', 'Set up cloud', and 'Dismiss'. A message about Jenkins running on Java 17, which ends support on March 31, 2026, with 'More Info' and 'Ignore' buttons. The 'System Configuration' section contains links for System, Tools, Nodes, Clouds, Plugins, and Appearance.

The screenshot shows the Jenkins plugin manager search results for 'owasp'. The sidebar on the left has links for 'Updates', 'Available plugins' (which is selected), 'Installed plugins', 'Advanced settings', and 'Download progress'. The main table lists three available plugins:

Install	Name	Released
<input checked="" type="checkbox"/>	OWASP Dependency-Check 5.6.1	1 mo 10 days ago
<input type="checkbox"/>	OWASP Dependency-Track 6.0.1	2 mo 8 days ago
<input type="checkbox"/>	Official OWASP ZAP 1.1.0	7 yr 10 mo ago
<input type="checkbox"/>	OWASP ZAP 1.0.7	7 yr 10 mo ago

Each plugin row includes a brief description and a note about its status or potential risks.

Go to manage jenkins and click on tools to add the Dependency check tool from github

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like 'New Item', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins' (which is selected), and 'My Views'. The main area has a title 'Manage Jenkins'. A prominent message says: 'Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#)'. Below this, another message states: 'Java 17 end of life in Jenkins'. It says: 'You are running Jenkins on Java 17, support for which will end on or after Mar 31, 2026. Refer to [the documentation](#) for more details.' There are 'More Info' and 'Ignore' buttons. The 'System Configuration' section includes links for 'System', 'Tools', 'Nodes', 'Clouds', 'Plugins', and 'Appearance'.

give the dependency check a name, click on “install automatically” and install from github

This screenshot shows the 'Dependency-Check installations' configuration screen. It has a header 'Maven installations' and a status 'Edited'. Below it, there's a 'Dependency-Check' section with a 'Name' field containing 'DP-Check'. Underneath, there's a checkbox 'Install automatically' which is checked. A dropdown menu is open under 'Add Installer' with the following options: 'Extract *.zip/*.tar.gz', 'Install from github.com' (which is highlighted in grey), 'Run Batch Command', and 'Run Shell Command'. At the bottom are 'Save' and 'Apply' buttons.

Install with latest or stable version



Next is to get the NVD key that the DP check scan would use for the scanning.
visit www.nvd.nist.gov fill in the form

To request an NVD API Key, please provide your organization name and a valid email address, and indicate your organization type. You must scroll to end of the Terms of Use Agreement and check "I agree to the Terms of Use" to obtain an API Key. Upon submitting the request, you will receive an email containing a single-use hyperlink that is used to activate and view your API Key. If your key is not activated within seven days, a new request for an API Key must be submitted.

Agree to the terms and condition and submit. You would get a notification via the email you registered it with.

The screenshot shows a web browser window with the URL nvd.nist.gov/developers/api-key-requested. The page displays a form for requesting an API key. The fields are as follows:

- Organization Name:** franklin Onyia (highlighted in red with the error message "must not be empty")
- Email Address:** onyiafranklin89@gmail.com (highlighted in red with the error message "must not be empty")
- Organization Type:** Personal Use / Not Listed

Below the form, a note states: "The API is provided "as is" and on an "as-available" basis. The NVD hereby disclaim all warranties of any kind, express or implied, including without limitation the warranties of merchantability, fitness for a particular purpose, and non-infringement. The NVD makes no warranty that the API will be error free or that access thereto will be continuous or uninterrupted."

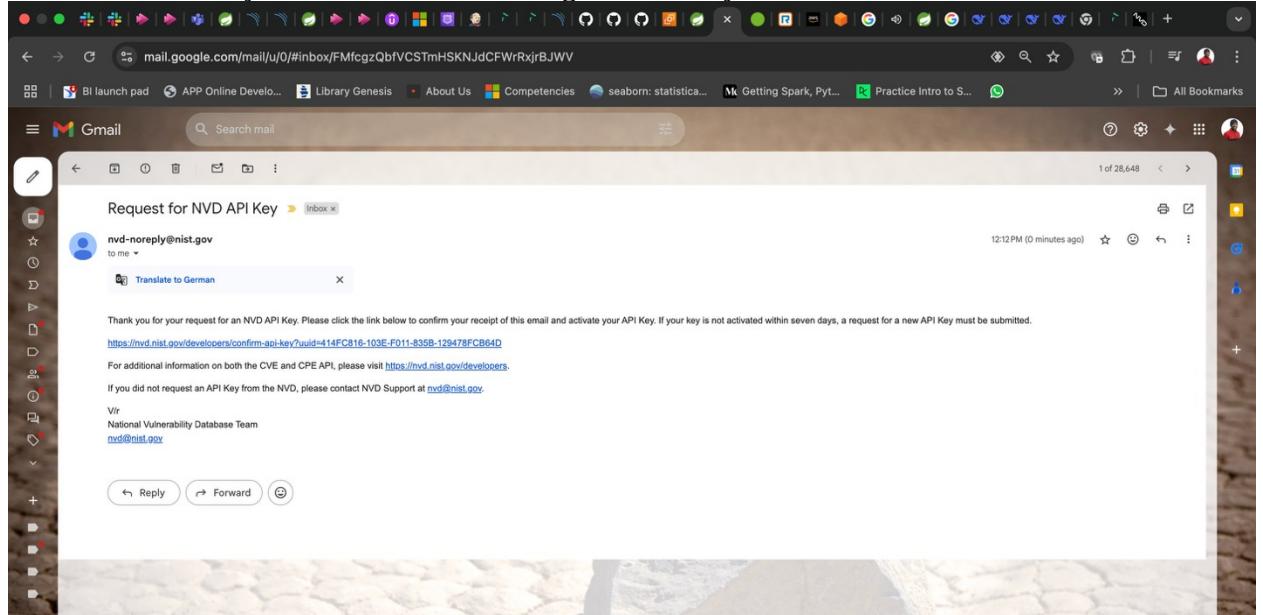
No Waiver

The NVD's failure to exercise or enforce any right or provision of these Terms of Use shall not constitute waiver of such right or provision.

I agree to the Terms of Use

This site is protected by reCAPTCHA and the Google Privacy Policy and Terms of Use apply.

Follow the link on your email and view the generated key from NVD



Copy the key

The screenshot shows a web browser window with the URL nvd.nist.gov/developers/confirm-api-key?uuuid=414FC816-103E-F011-835B-129478FCB64D. The page header includes the NIST logo and the text "Information Technology Laboratory" and "NATIONAL VULNERABILITY DATABASE". A banner at the top states "An official website of the United States government [Here's how you know](#)". The main content area displays the message "NVD API Key Activated" and provides instructions for saving the API key. It also shows the API key value: "API Key: baabfc6c-e909-4f77-bde7-76865e3893a4". Below this, it says "To request a new API key, please resubmit an API Key Request. Please note that activating a new key will deactivate the key shown above."

Copy the ket and use it to create a secret text credentials for the nvd key .
go to manage jenkins a and click on credentials to add the credentials.

The screenshot shows a web browser window with the URL jenkins.edenboutique.space/manage/. The page title is "Manage Jenkins". The left sidebar shows "Dashboard > Manage Jenkins". The main content area is titled "System Configuration" and contains several configuration sections: "System" (Configure global settings and paths), "Tools" (Configure tools, their locations and automatic installers), "Nodes" (Add, remove, control and monitor the various nodes that Jenkins runs jobs on), "Clouds" (Add, remove, and configure cloud instances to provision agents on-demand), "Security" (Secure Jenkins; define who is allowed to access/use the system), "Credentials" (Configure credentials), "Users" (Create/delete/modify users that can log in to this Jenkins), and "Appearance" (Configure the look and feel of Jenkins). A status message at the top right says: "You are running Jenkins on Java 17, support for which will end on or after Mar 31, 2026. Refer to [the documentation](#) for more details."

Give the credential a name and paste the key

New credentials

Kind

Secret text

Scope ?
Global (Jenkins, nodes, items, all child items, etc)

Secret
.....

ID ?
nvd-key

Description ?
nvd-key

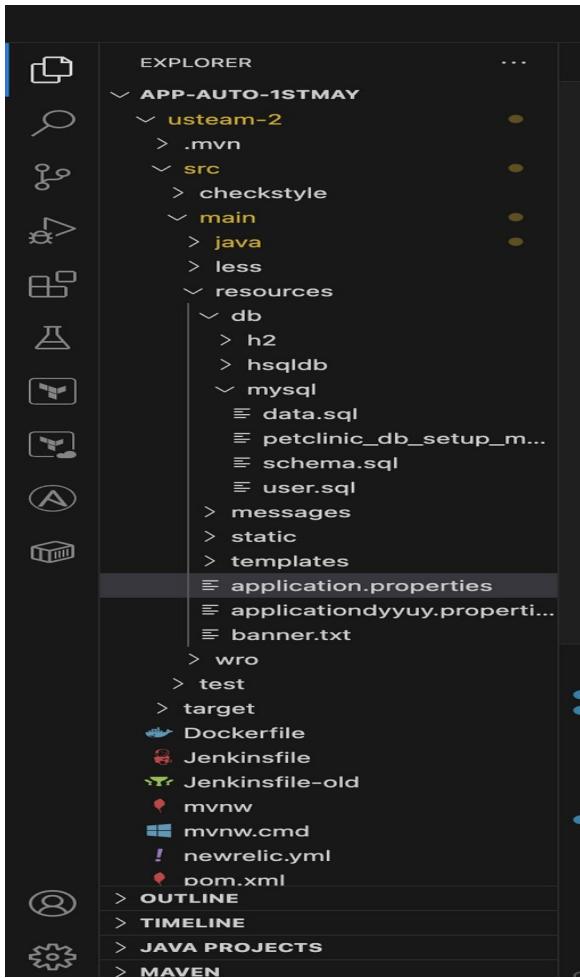
Create

View all credentials created

ID	Name	Kind	Description
slack	slack	Secret text	slack
git-cred	onyiafranklin/******** (git-cred)	Username with password	git-cred
ansible-key	ec2-user (ansible-key)	SSH Username with private key	ansible-key
sonarqube	sonarqube	Secret text	sonarqube
nexus-cred	admin/******** (nexus-cred)	Username with password	nexus-cred
nexus-username	nexus-username	Secret text	nexus-username
nexus-password	nexus-password	Secret text	nexus-password
nexus-docker-repo	nexus-docker-repo	Secret text	nexus-docker-repo
bastion-ip	bastion-ip	Secret text	bastion-ip
ansible-ip	ansible-ip	Secret text	ansible-ip
nvd-key	nvd-key	Secret text	nvd-key

Next is to go to the application code on vscode to set the mysql for your application to persist data on the created database.

On the application code, go to the “application.properties” to set the database url endpoint and database name.



we set the db endpoint and db name on line 3

```

# database init, supports mysql too
database=mysql
spring.datasource.url=$MYSQL_URL:jdbc:mysql://auto-discovery-db.ct2yoke8a24e.eu-west-2.rds.amazonaws.com:3306/petadoption
spring.datasource.username=$MYSQL_USER:petclinic
spring.datasource.password=$MYSQL_PASS:petclinic
spring.datasource.schemaclasspath:db/$database/schema.sql
spring.datasource.dataclasspath:db/$database/data.sql
# SQL is written to be idempotent so this is safe
spring.datasource.initialization-mode=always

```

On the mysql, make sure you the database name exist on the user.sql

```
CREATE DATABASE IF NOT EXISTS petadoption;
ALTER DATABASE petadoption
  DEFAULT CHARACTER SET utf8
  DEFAULT COLLATE utf8_general_ci;
GRANT ALL PRIVILEGES ON petadoption.* TO 'petclinic'@'%' IDENTIFIED BY 'petclinic';
```

also check the database schema

```
CREATE TABLE IF NOT EXISTS vets (
  id INT(4) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  first_name VARCHAR(30),
  last_name VARCHAR(30),
  INDEX(last_name)
) engine=InnoDB;

CREATE TABLE IF NOT EXISTS specialties (
  id INT(4) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(80),
  INDEX(name)
) engine=InnoDB;

CREATE TABLE IF NOT EXISTS vet_specialties (
  vet_id INT(4) UNSIGNED NOT NULL,
  specialty_id INT(4) UNSIGNED NOT NULL,
  FOREIGN KEY (vet_id) REFERENCES vets(id),
  FOREIGN KEY (specialty_id) REFERENCES specialties(id),
  UNIQUE (vet_id,specialty_id)
) engine=InnoDB;

CREATE TABLE IF NOT EXISTS types (
  id INT(4) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(80),
  INDEX(name)
) engine=InnoDB;
```

also view the previous data on data.sql

The screenshot shows a code editor interface with several tabs open. The tabs include Jenkinsfile, user.sql, data.sql (which is currently selected), and application.properties. The data.sql tab contains a series of SQL INSERT statements for populating a MySQL database with data related to vets, specialties, and types.

```

1  INSERT IGNORE INTO vets VALUES (1, 'James', 'Carter');
2  INSERT IGNORE INTO vets VALUES (2, 'Helen', 'Leary');
3  INSERT IGNORE INTO vets VALUES (3, 'Linda', 'Douglas');
4  INSERT IGNORE INTO vets VALUES (4, 'Rafael', 'Ortega');
5  INSERT IGNORE INTO vets VALUES (5, 'Henry', 'Stevens');
6  INSERT IGNORE INTO vets VALUES (6, 'Sharon', 'Jenkins');
7
8  INSERT IGNORE INTO specialties VALUES (1, 'radiology');
9  INSERT IGNORE INTO specialties VALUES (2, 'surgery');
10 INSERT IGNORE INTO specialties VALUES (3, 'dentistry');
11
12 INSERT IGNORE INTO vet_specialties VALUES (2, 1);
13 INSERT IGNORE INTO vet_specialties VALUES (3, 2);
14 INSERT IGNORE INTO vet_specialties VALUES (3, 3);
15 INSERT IGNORE INTO vet_specialties VALUES (4, 2);
16 INSERT IGNORE INTO vet_specialties VALUES (5, 1);
17
18 INSERT IGNORE INTO types VALUES (1, 'cat');
19 INSERT IGNORE INTO types VALUES (2, 'dog');
20 INSERT IGNORE INTO types VALUES (3, 'lizard');
21 INSERT IGNORE INTO types VALUES (4, 'snake');
22 INSERT IGNORE INTO types VALUES (5, 'bird');
23 INSERT IGNORE INTO types VALUES (6, 'hamster');
24
25 INSERT IGNORE INTO owners VALUES (1, 'George', 'Franklin', '110 W. Liberty St.', 'Madison', '6085551023');
26 INSERT IGNORE INTO owners VALUES (2, 'Betty', 'Davis', '638 Cardinal Ave.', 'Sun Prairie', '6085551749');
27 INSERT IGNORE INTO owners VALUES (3, 'Eduardo', 'Rodriguez', '2693 Commerce St.', 'McFarland', '6085558763');

```

Below the code editor, a terminal window shows a git commit message:

```

franklinonyla@MACBOOKPRO-CEA1:~/steam-2$ git add .
franklinonyla@MACBOOKPRO-CEA1:~/steam-2$ git commit -m "update"
[eu-team2-app 77ee07d] update
 8 files changed, 68 insertions(+), 64 deletions(-)
 delete mode 100644 src/main/resources/application-mysql.properties
 delete mode 100644 target/classes/application-mysql.properties
 create mode 100644 target/classes/application-dvyyu.properties

```

Now is time to create the application pipeline
click on “new item”

The screenshot shows the Jenkins dashboard. In the center, there is a table listing pipelines. One pipeline, named "infra-pipeline", is highlighted with a green checkmark icon. The table columns are S (Status), W (Workflow), Name (infra-pipeline), Last Success (56 min), Last Failure (N/A), and Last Duration (5 min 0 sec).

S	W	Name	Last Success	Last Failure	Last Duration
		infra-pipeline	56 min #2	N/A	5 min 0 sec

Below the table, there is a "Build Queue" section which is currently empty. The status bar at the bottom indicates "Icon: S M L".

Give your new pipeline a name

New Item

Enter an item name
app-pipeline

Select an item type

Pipeline Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Freestyle project Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Maven project Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Multi-configuration project Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK

Copy the url of the application code

github.com/onyiafranklin/usteam-2/tree/eu-team2-app

usteam-2 Public
forked from CloudHigh/usteam

This branch is 4 commits ahead of CloudHigh/usteam:eu-team2-app .

onyiafranklin test

.mvn(wrapper) first commit

src update eu team 2 a

target test

Dockerfile update

Jenkinsfile update

Local Codespaces

Clone

HTTPS SSH GitHub CLI

<https://github.com/onyiafranklin/usteam-2>

Clone using the web URL.

Open with GitHub Desktop

Download ZIP

2 hours ago

About

No description, website, or topics provided.

Readme

Activity

0 stars

0 watching

0 forks

Releases

No releases published

Create a new release

add the url to the repository url and add your git credentials

The screenshot shows the Jenkins Pipeline configuration page. On the left, a sidebar lists 'General', 'Triggers', 'Pipeline' (which is selected), and 'Advanced'. The main area is titled 'Pipeline' with the sub-section 'Definition'. It says 'Define your Pipeline using Groovy directly or pull it from source control.' A dropdown menu shows 'Pipeline script from SCM'. Below this, a section for 'SCM' is expanded, showing 'Git' selected. Under 'Repositories', there is one entry with 'Repository URL' set to 'https://github.com/onyiafranklin/usteam-2.git' and 'Credentials' set to 'onyiafranklin/******** (git-cred)'. There is also a '+ Add' button and an 'Advanced' dropdown. At the bottom are 'Save' and 'Apply' buttons.

add the branch you would like to build and add the script path you would like to build which is Jenkinsfile

save and apply to see the pipeline compile

The screenshot shows the Jenkins Pipeline configuration page. The 'Pipeline' section is expanded, showing 'Branches to build'. Under 'Branch Specifier (blank for 'any')', the value is '/eu-team2-app'. There is an 'Add Branch' button. Below this is 'Repository browser' set to '(Auto)'. Under 'Additional Behaviours', there is an 'Add' button. At the bottom is a 'Script Path' field containing 'Jenkinsfile' and a checked checkbox for 'Lightweight checkout'. At the very bottom are 'Save' and 'Apply' buttons.

```

management capabilities on your system.
View your connected systems at https://console.redhat.com/insights

You can learn more about how to register your system
using rhc at https://red.ht/registration
[ec2-user@Nexus ~]$ sudo cat /app/sonatype-work/nexus3/admin.password
0a300d52-1efa-41ce-af7d-74ca3d65638d[ec2-user@Nexus ~]$ exit
logout
Connection to 10.0.1.156 closed.
[ec2-user@bastion ~]$ exit
logout
Connection to 18.130.51.153 closed.
[ec2-user@jenkins infra-pipeline]$ sudo systemctl restart docker
[ec2-user@jenkins infra-pipeline]$ 

```

Ln 1, Col 27 (26 selected) Spaces: 4

watch the pipeline run

app-pipeline

Stage View

Declarative: Checkout SCM	Code analysis stage
649ms	52s

Average stage times:

Permalinks

Builds

No builds

Today

#1 11:22 AM

before deploy to production, it should request for approval.
thisnis important because ches such as user acceptability test, load test performance, security and compliance testing,as well as smoke test is done before approval to production environment

jenkins.edenboutique.space/job/app-pipeline/

Jenkins

Dashboard > app-pipeline >

app-pipeline

Dependency check	Build artefacts	Push artifacts to nexus-repo	Build Docker image	Login to Nexus repo	Push image to Nexus repo	Trivy image scan	Deploy to stage	check stage website availability	Request for Approval	Deploy to prod	check prod website availability
52s	37s	31s	13s	314ms	8s	23s	19s	5min 31s	89ms	2s	2s
19s	16s	1s	9s	319ms	5s	18s	19s	3min 22s	157ms (paused for 8s)	19s	17s

jenkins.edenboutique.space/job/app-pipeline/

Jenkins

Dashboard > app-pipeline >

app-pipeline

Declarative: Checkout SCM	Code analysis stage	Quality gate	Dependency check	Build artefacts	Push artifacts to nexus-repo	Build Docker image	Login to Nexus repo	Push image to Nexus repo	Trivy image scan	Deploy to stage	check stage website availability	Request for Approval	Deploy to prod	check prod website availability
498ms	23s	160ms	52s	37s	31s	13s	314ms	8s	23s	19s	5min 31s	89ms	2s	28s
411ms	12s	147ms (paused for 1s)	19s	16s	1s	9s	319ms	5s	18s	19s	3min 22s	157ms (paused for 8s)	19s	3min 22s

View deployed application using the stage url (stage.edenboutique.space)

stage.edenboutique.space

BI launch pad APP Online Develo... Library Genesis About Us Competencies seaborn: statistica... Getting Spark, Pyt... Practice Intro to S... All Bookmarks

 HOME FIND OWNERS VETERINARIANS ERROR

??Welcome to my Pet Adoption School_en_US??



spring Pivotal.

View deployed application using the prod url (www.edenboutique.space)

edenboutique.space

BI launch pad APP Online Develo... Library Genesis About Us Competencies seaborn: statistica... Getting Spark, Pyt... Practice Intro to S... All Bookmarks

 HOME FIND OWNERS VETERINARIANS ERROR

??Welcome to my Pet Adoption School_en_US??



spring Pivotal.

view quality gate scan results

QUALITY GATE STATUS

MEASURES

New Code	Overall Code
Since May 31, 2025 Started 7 hours ago	

New Bugs Reliability **A**

New Vulnerabilities Security **A**

New Security Hotspots Security Review **A**

Added Debt **New Code Smells** Maintainability **A**

Coverage on 0 New Lines to cover

Duplications on 0 New Lines

view dependency check scan results

Dependency-Check Results

SEVERITY DISTRIBUTION

Severity	Count
Critical	38
High	92
Medium	17
Low	12

Status

- </> Changes
- Console Output
- Edit Build Information
- Delete build #8*
- Timings
- Git Build Data
- Dependency-Check**
- Pipeline Overview
- Restart from Stage
- Replay
- Pipeline Steps
- Workspaces
- ← Previous Build

File Name	Vulnerability	Severity	Weakness
+ bootstrap-3.3.6.jar	NVD CVE-2016-10735	Medium	CWE-79
+ bootstrap-3.3.6.jar	NVD CVE-2018-14041	Medium	CWE-79
+ bootstrap-3.3.6.jar	NVD CVE-2018-14042	Medium	CWE-79
+ bootstrap-3.3.6.jar	NVD CVE-2018-20676	Medium	CWE-79
+ bootstrap-3.3.6.jar	OSSINDEX CVE-2018-14040	Medium	CWE-79
+ bootstrap-3.3.6.jar	OSSINDEX CVE-2018-20677	Medium	CWE-79
+ bootstrap-3.3.6.jar	OSSINDEX CVE-2019-8331	Medium	CWE-79
+ bootstrap-3.3.6.jar	OSSINDEX CVE-2024-6484	Medium	CWE-79
+ bootstrap-3.3.6.jar	OSSINDEX CVE-2024-6531	Medium	CWE-79
+ bootstrap-3.3.6.jar	OSSINDEX CVE-2024-6485	Low	CWE-79

K. PERSISTING OUR DATA TO OUR DATABASE

Now lets check if our database persisted data

login to the stage server through the bastion host

```
-rw-r--r--. 1 root jenkins 300 May 31 10:18 variable.tf
drwxr-xr-x. 2 root jenkins 127 May 31 10:14 vault-jenkins
[ec2-user@jenkins infra-pipeline]$ ssh -i auto-discov-key.pem ec2-user@18.130.51.153
Register this system with Red Hat Insights: insights-client --register
Create an account or view all your systems at https://red.ht/insights-dashboard
Last login: Sat May 31 10:48:53 2025 from 18.170.218.148
[ec2-user@bastion ~]$ ssh 10.0.4.220
The authenticity of host '10.0.4.220 (10.0.4.220)' can't be established.
ED25519 key fingerprint is SHA256:iKp473mQBUhVYobcpVqPw8AXhaCDUDgGv/yCUVG+wX0.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.4.220' (ED25519) to the list of known hosts.
Register this system with Red Hat Insights: rhc connect

Example:
# rhc connect --activation-key <key> --organization <org>

The rhc client and Red Hat Insights will enable analytics and additional
management capabilities on your system.
View your connected systems at https://console.redhat.com/insights

You can learn more about how to register your system
using rhc at https://red.ht/registration
Last login: Sat May 31 17:01:29 2025
[ec2-user@stage-instance ~]$
```

by onyiafranklin (1 week ago) L

Install mysql

```
You can learn more about how to register your system
using rhc at https://red.ht/registration
Last login: Sat May 31 17:01:29 2025
[ec2-user@stage-instance ~]$ sudo yum install mysql -y
Updating Subscription Management repositories.
Unable to read consumer identity

This system is not registered with an entitlement server. You can use "rhc" or "subscription-manager" to register.
```

Last metadata expiration check: 1:59:16 ago on Sat 31 May 2025 03:13:01 PM UTC.
Dependencies resolved.

Package	Architecture	Version	Repository
---------	--------------	---------	------------

login to the database

```
Complete!
[ec2-user@stage-instance ~]$ mysql -h auto-discov-db.ct2yoke8a24e.eu-west-2.rds.amazonaws.com -u petclinic -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 149
Server version: 5.7.44-rds.20250213 MySQL 5.7 is now under RDS Extended Support. Please upgrade to 8.0 to benefit from the latest RDS innovation

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>
```

show database

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 149
Server version: 5.7.44-rds.20250213 MySQL 5.7 is now under RDS Extended Support. Please upgrade to 8.0 to benefit from the latest RDS innov
Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| innodb |
| mysql |
| performance_schema |
| petadoption |
| sys |
+-----+
6 rows in set (0.00 sec)

mysql> 

```

φ onviafranklin (1 week ago) Ln 91, Col 12 Spaces: 4 UTF-8

```
show tables
6 rows in set (0.00 sec)

mysql> use petadoption
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_petadoption |
+-----+
| owners |
| pets |
| specialties |
| types |
| vet_specialties |
| vets |
| visits |
+-----+
7 rows in set (0.00 sec)

mysql> 

```

φ onviafranklin (1 week ago)

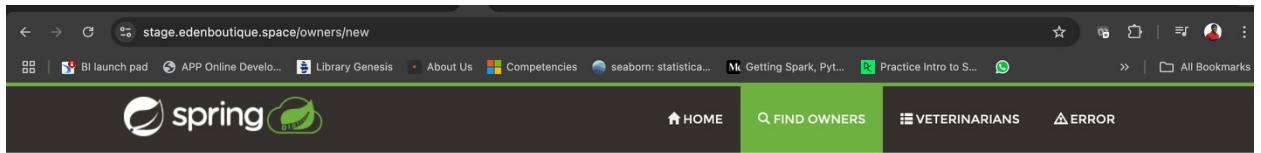
```
select to see owners table
+-----+
| vets |
| visits |
+-----+
7 rows in set (0.00 sec)

mysql> select * from owners
-> ;
+-----+-----+-----+-----+-----+
| id | first_name | last_name | address | city | telephone |
+-----+-----+-----+-----+-----+
| 1 | George | Franklin | 110 W. Liberty St. | Madison | 6085551023 |
| 2 | Betty | Davis | 638 Cardinal Ave. | Sun Prairie | 6085551749 |
| 3 | Eduardo | Rodriguez | 2693 Commerce St. | McFarland | 6085558763 |
| 4 | Harold | Davis | 563 Friendly St. | Windsor | 6085553198 |
| 5 | Peter | McTavish | 2387 S. Fair Way | Madison | 6085552765 |
| 6 | Jean | Coleman | 105 N. Lake St. | Monona | 6085552654 |
| 7 | Jeff | Black | 1450 Oak Blvd. | Monona | 6085555387 |
| 8 | Maria | Escobito | 345 Maple St. | Madison | 6085557683 |
| 9 | David | Schroeder | 2749 Blackhawk Trail | Madison | 6085559435 |
| 10 | Carlos | Estaban | 2335 Independence La. | Waunakee | 6085555487 |
+-----+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql> 

```

Now lets add data to the database and see if its persisted on the database
add data as shown below



Owner

First Name	richard	✓
Last Name	olawole	✓
Address	17 bello close chevyview estate lekki lagos	✓
City	LEKKI	✓
Telephone	0703579745	✓

[Add Owner](#)



Owner Information

Name	richard olawole
Address	17 bello close chevyview estate lekki lagos
City	LEKKI
Telephone	0703579745

[Edit Owner](#) [Add New Pet](#)

Pets and Visits



select * from table to see Richard >> firstname and olawole>> last name added has been persisted

```

+-----+-----+-----+-----+-----+
| id   | first_name | last_name | address           | city    | telephone |
+-----+-----+-----+-----+-----+
| 1    | George     | Franklin  | 110 W. Liberty St. | Madison | 6085551023 |
| 2    | Betty      | Davis     | 638 Cardinal Ave. | Sun Prairie | 6085551749 |
| 3    | Eduardo    | Rodriguez | 2693 Commerce St. | McFarland | 6085558763 |
| 4    | Harold     | Davis     | 563 Friendly St.  | Windsor  | 6085553198 |
| 5    | Peter      | McTavish  | 2387 S. Fair Way  | Madison  | 6085552765 |
| 6    | Jean       | Coleman   | 105 N. Lake St.   | Monona   | 6085552654 |
| 7    | Jeff       | Black     | 1450 Oak Blvd.    | Monona   | 6085555387 |
| 8    | Maria      | Escobito  | 345 Maple St.     | Madison  | 6085557683 |
| 9    | David      | Schroeder | 2749 Blackhawk Trail | Madison | 6085559435 |
| 10   | Carlos     | Estaban   | 2335 Independence La. | Waunakee | 6085555487 |
| 11   | richard   | olawole   | 17 bello close chevyview estate lekki lagos | LEKKI | 0703579745 |
+-----+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql> select * from owners;
+-----+-----+-----+-----+-----+
| id   | first_name | last_name | address           | city    | telephone |
+-----+-----+-----+-----+-----+
| 1    | George     | Franklin  | 110 W. Liberty St. | Madison | 6085551023 |
| 2    | Betty      | Davis     | 638 Cardinal Ave. | Sun Prairie | 6085551749 |
| 3    | Eduardo    | Rodriguez | 2693 Commerce St. | McFarland | 6085558763 |
| 4    | Harold     | Davis     | 563 Friendly St.  | Windsor  | 6085553198 |
| 5    | Peter      | McTavish  | 2387 S. Fair Way  | Madison  | 6085552765 |
| 6    | Jean       | Coleman   | 105 N. Lake St.   | Monona   | 6085552654 |
| 7    | Jeff       | Black     | 1450 Oak Blvd.    | Monona   | 6085555387 |
| 8    | Maria      | Escobito  | 345 Maple St.     | Madison  | 6085557683 |
| 9    | David      | Schroeder | 2749 Blackhawk Trail | Madison | 6085559435 |
| 10   | Carlos     | Estaban   | 2335 Independence La. | Waunakee | 6085555487 |
| 11   | richard   | olawole   | 17 bello close chevyview estate lekki lagos | LEKKI | 0703579745 |
+-----+-----+-----+-----+-----+
11 rows in set (0.00 sec)

mysql> ■

```

edenboutique.space

spring

HOME FIND OWNERS VETERINARIANS ERROR

??Welcome to my Pet Adoption School_en_US??

spring

L. CHANGE ON APPLICATION PIPELINE AND REDEPLOYMENT TO VIEW CHANGE

Now lets change the welcome page tag name to (welcome to Ebuka's adoption and run

the pipeline again) and run the pipeline again.

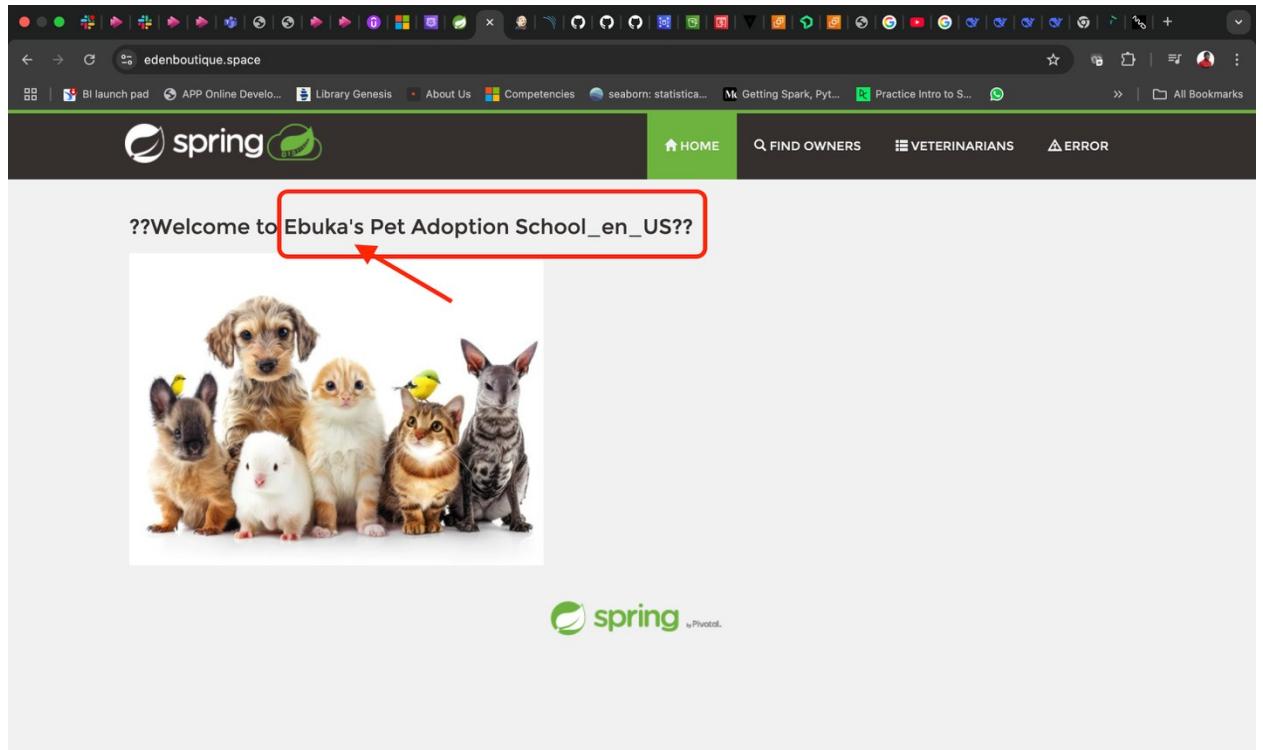
The screenshot displays a developer's environment with multiple windows open:

- Code Editor:** Shows a Jenkinsfile and a welcome.html file. The welcome.html file contains Thymeleaf code with a highlighted section:

```
<h1 th:text="Welcome to Ebuka's Adoption School">Welcome to Ebuka's Adoption School</h1>
```
- Browser Preview:** A screenshot of a web browser showing the rendered content of the welcome.html file.
- Jenkins Pipeline Status:** The Jenkins interface shows the "app-pipeline" job. The pipeline stages are listed on the left: Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, SonarQube, Stages, Rename, and Pipeline Syntax. The main area shows the pipeline execution timeline from May 31, 19:10 to May 31, 17:58. Average stage times are listed above each stage. A dependency check trend chart is displayed on the right, showing a sharp increase in dependency count from stage #1 to #2, peaking at approximately 178 before stabilizing.

Stage	Average Stage Time (ms)	Start Time	End Time
Checkout SCM	500ms	May 31, 19:10	May 31, 19:10
Code analysis stage	20s	May 31, 19:10	May 31, 19:10
Quality gate	157ms	May 31, 19:10	May 31, 19:10
Dependency check	45s	May 31, 19:10	May 31, 19:10
Build artifacts	31s	May 31, 19:10	May 31, 19:10
Push artifacts to nexus-repo	38s	May 31, 19:10	May 31, 19:10
Build Docker image	12s	May 31, 19:10	May 31, 19:10
Login to Nexus repo	313ms	May 31, 19:10	May 31, 19:10
Push image to Nexus repo	7s	May 31, 19:10	May 31, 19:10
Trivy image scan	22s	May 31, 19:10	May 31, 19:10
Deploy to stage	19s	May 31, 19:10	May 31, 19:10
check stage website availability	5min 2s	May 31, 19:10	May 31, 19:10
Request for Approval	9min 2s	May 31, 19:10	May 31, 19:10
Deploy to prod	16s	May 31, 19:10	May 31, 19:10
check prod website availability	50s	May 31, 19:10	May 31, 19:10

the change was effected



M. MONITORING WITH NEWRELIC

Application and infrastructure monitoring is very important to avoid disruption of service.

newrelic agent were installed on all server for monitoring and newrelic agent was also installed for application monitoring.

results of monitoring.

view all servers monitored showing the cpu usage, memory

The screenshot shows the New Relic interface for monitoring entities. The left sidebar is titled "All Entities" and lists various monitoring categories: Quick Find, Integrations & Agents, All Capabilities, All Entities (selected), Dashboards, APM & Services, Logs, Traces, Synthetic Monitoring, Alerts, Infrastructure, Kubernetes, Browser, Mobile, Errors Inbox, Apps, Help, Add User, Upgrade Now, and franklin. The main content area is titled "All Entities" and shows two tables: "Services - APM" and "Hosts".

Services - APM

Name	Response time	Throughput	Error rate
petclinicapps	28 ms	8.37 rpm	0%
SupportChallengeapp	-	-	-

Hosts

Name	Agent ...	CPU us...	Memor...	Storag...	Networ...	Networ...
ansible-server	1.64.0	9.02%	18.25%	21.25%	1.2 kB/s	1.14 kB...
prod-instance	1.64.0	3.77%	29.15%	34.09%	992 B/s	773 B/s...
stage-instance	1.64.0	4.91%	29.53%	34.39%	924 B/s	1.62 kB...
ansible-server	1.64.0	-	-	-	-	-

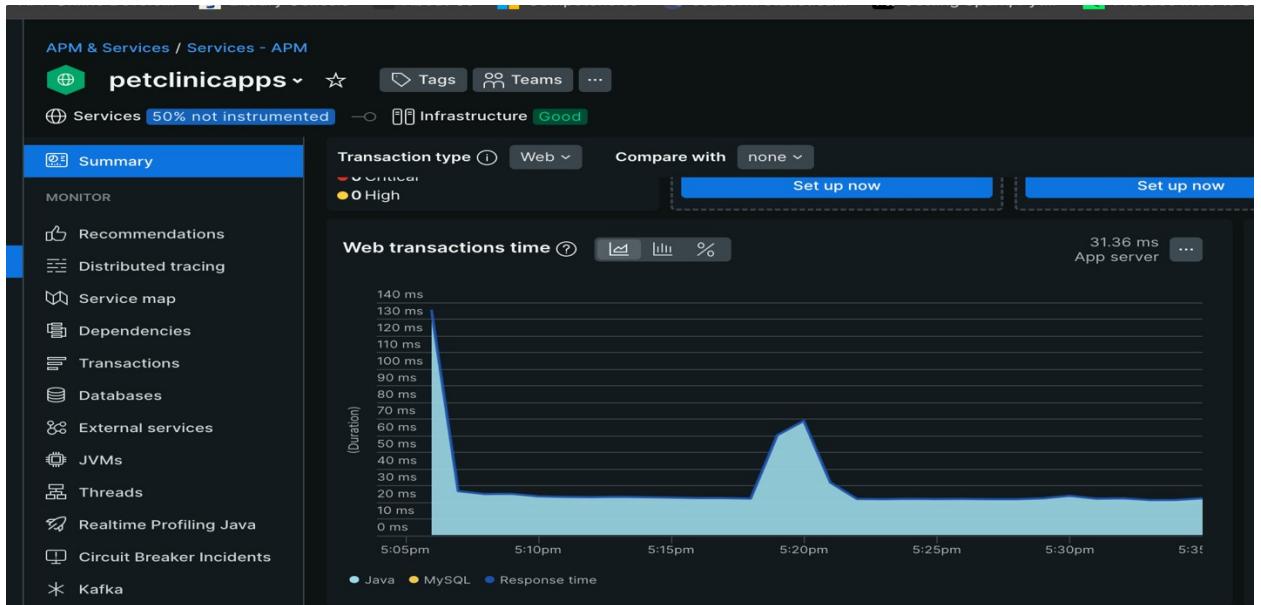
see application monitoring .it shows respons time, throughput and error rate

The screenshot shows the New Relic interface for APM & Services monitoring. The left sidebar is titled "APM & Services" and lists: Quick Find, Integrations & Agents, All Capabilities, All Entities, Dashboards, APM & Services (selected), and Logs. The main content area is titled "Entities" and shows a single table.

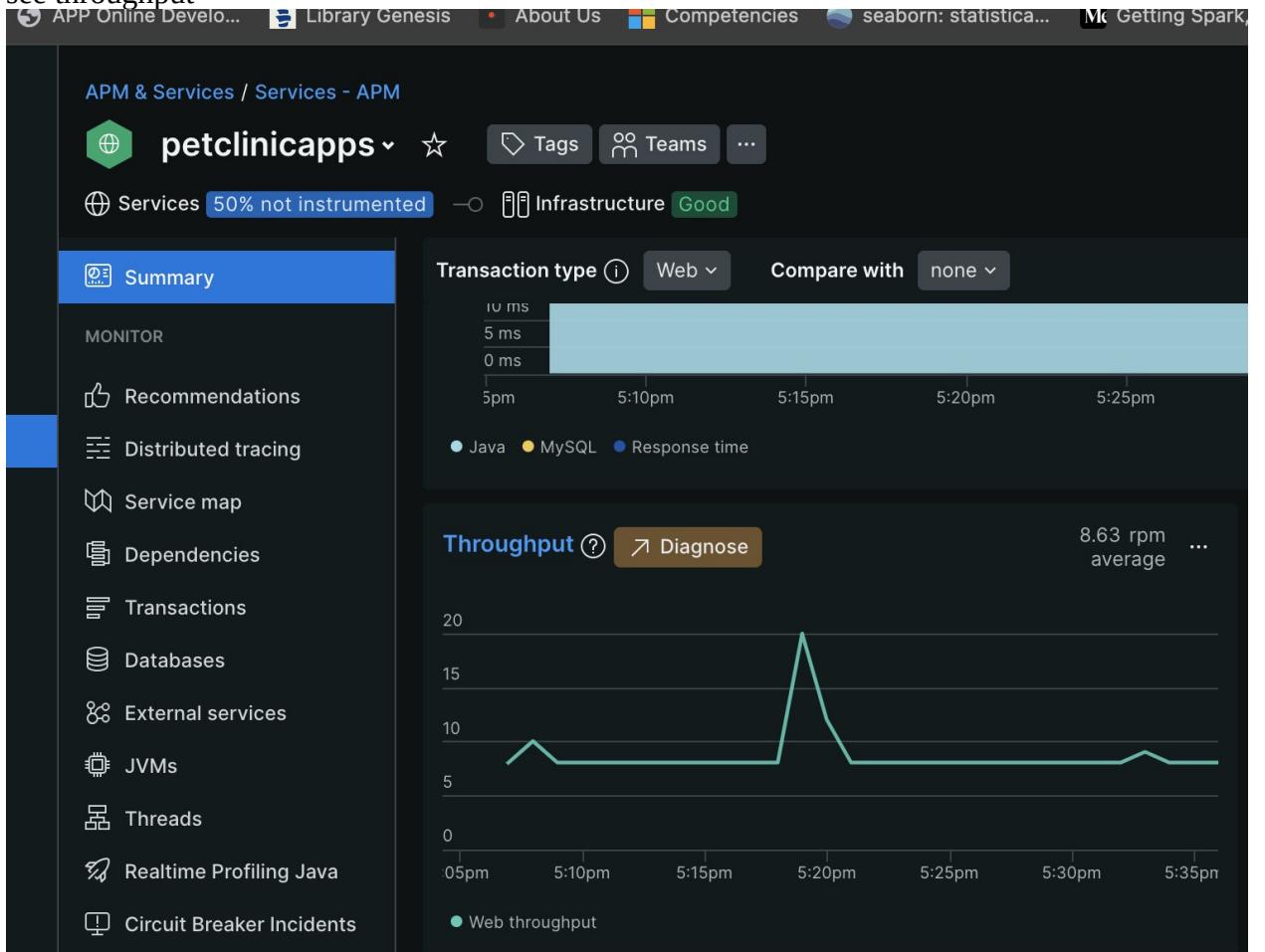
Entities

Name	Response time	Throughput	Error rate
petclinicapps	28 ms	8.37 rpm	0%

see web transaction time



see throughput



see infrastructure monitoring

Hosts Metrics

Overview: Infrastructure / Hosts

Metrics:

- CPU (%): Shows a sharp peak at 5:00pm.
- Memory used (%): Shows a steady increase from ~15% to ~25%.
- Disk used (%): Shows a constant baseline around 35%.
- Disk utilization (%): Shows a sharp peak at 5:00pm.

Summary:

View	Name	Applications	Agent version	CPU usage (%)	Memory usage (%)	Storage usage (%)
●	ansible-server	--	1.64.0	8.93%	18.23%	21.2 ...
●	prod-instance	--	1.64.0	3.79%	29.14%	34.0 ...
●	stage-instance	--	1.64.0	4.89%	29.58%	34.4 ...

APM & Services - APM

petclinicapps - 50% not instrumented

Summary:

Issues: 0 Critical, 0 High

Deployments: Set up now

Service levels: Set up now

Vulnerabilities: 8 Critical, 27 High

Web transactions time: 31.36 ms App server

Apdex score: 1 [0.5] App Server

Duration: 0 ms to 350 µs

Java MySQL Response time

Instances: All instances

one.eu.newrelic.com/nr1-core/apm/overview/NjQ5NjM0MnxBUE18QVBQTEIDQVRJT058MzY3ODQ4NzA3?account=6496342&duration=1800000&st...

Services 50% not instrumented Infrastructure Good

Summary Transaction type Web Compare with none Instances All instances

Issues ① Deployments ① Service levels ① Vulnerabilities ①

Java MySQL Response time

Web transactions time 31.36 ms App server

Apdex score 1 [0.5] App Server

Throughput 8.63 rpm average

Errors 0 % average

Java MySQL Response time

Throughput 8.63 rpm average

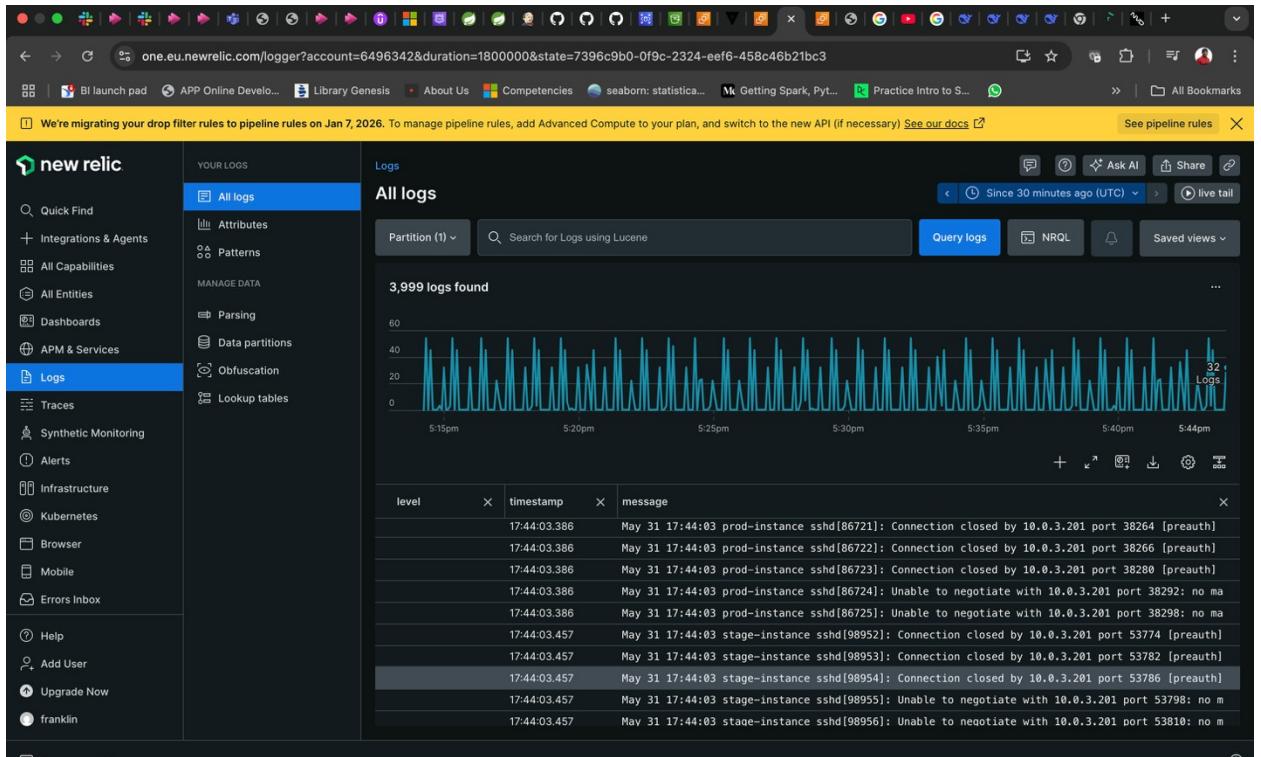
Errors 0 % average

Java MySQL Response time

Transactions 5 slowest transactions (by total time)

Transaction name	Slowest trace	Error rate	Average dura...
owners/find (GET)	--	0%	1.22 s
owners/new (POST)	--	0%	197 ms

Query your data



Further improvement:

Working to automate access key for vault to be copied automatically into the vault provider file
tighten up security by shutting ssh port and granting access through SSM only

Conclusion

In conclusion, automating application deployment is essential for organizations to meet evolving market demands efficiently. By streamlining deployment processes, businesses can accelerate delivery, enhance productivity, and ultimately drive greater profitability.