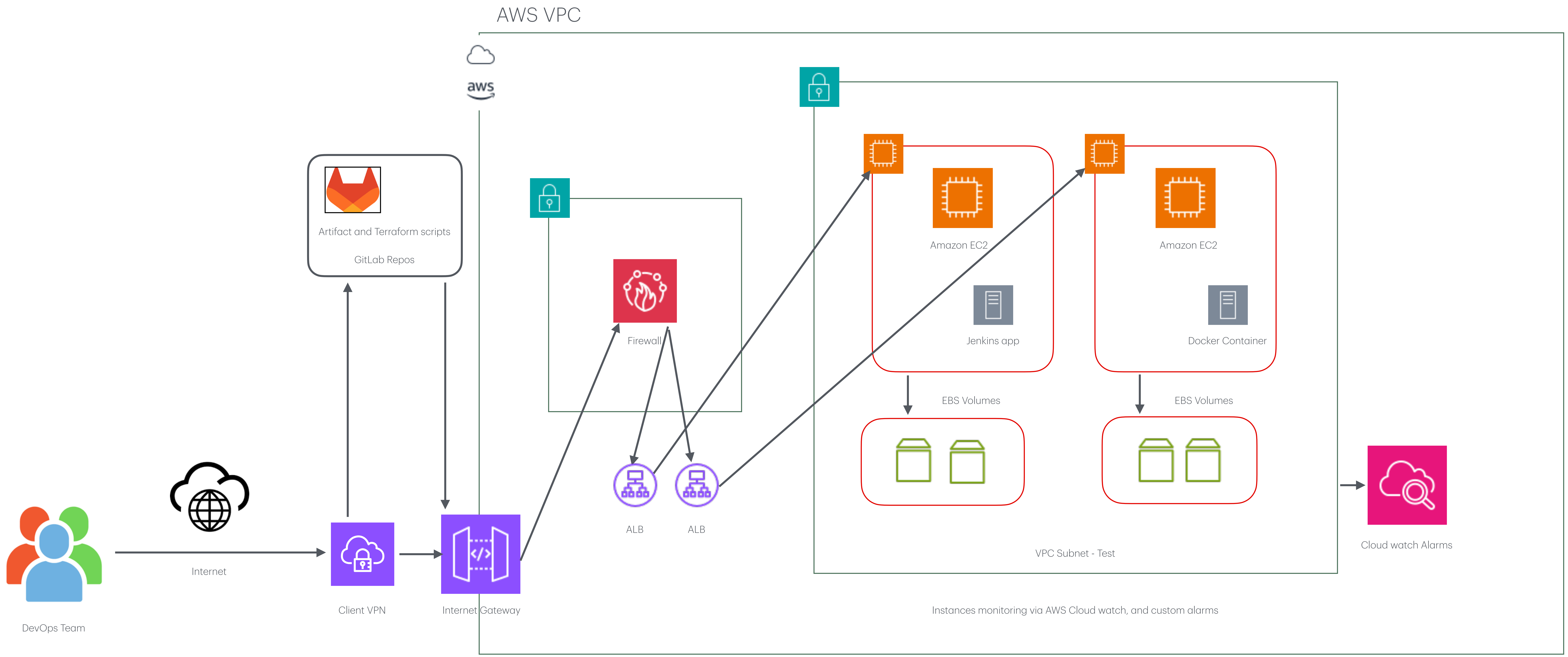
BH

High-Level Architecture



Architecture

Componentes

	Target users
	Internet outside of VPC (Virtual Private Cloud)
aws	Virtual Private Cloud (Amazon VPC), you can launch AWS resources in a logically isolated virtual network that you've defined
	Client VPN is a managed client-based VPN service that enables you to securely access your AWS resources and resources in your on-premises network
	Internet Gateway is an AWS component that provides a path for network traffic to travel between a Virtual Private Cloud (VPC) and the public internet
	Git repository is a central storage location for managing and tracking changes in files and directories
	A subnet is a range of IP addresses in your VPC. You can create AWS resources, such as EC2 instances, in specific subnets
	Network traffic filter
	Application Load Balancer routes traffic to targets (EC2 instances, containers, IP addresses, and others) based on the content of the request
	Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud

Architecture

Componentes

	EBS volume is a durable, block-level storage device attached to the Ec2 Instances
	Represent the content of the app server
	Amazon CloudWatch monitors the Amazon Web Services (AWS) resources and the applications in real time

Instructions

PRE REQUISITES

- Pre Requisites
  - AWS Account
  - Terraform client (Mac OS)
      - Make sure terraform is working executing the following command:
        - $ terraform version
          - The output should be similar to this (Terraform v1.9.5)
  - AWS CLI package (Mac OS)
    - Make sure the local was cli profile is proper configured. Terraform will look into the aws local profile to run.
  - GIt Client to access to Git Repos (AWS Keys, Terraform Scrips, Artifacts Repos) to pull and push.
- Note: Here is presented two simple ways
  - 1. Assuming there is no infra - running Terraform form the local PC - The high-level architecture / instructions represents this item.
  - 2. Assuming infra is in place and Jenkins is been used already - Only the pipeline file to deploy the assets will be provided.

Instructions

Create a AWS key pair

1. Access AWS Account
   1. On Services Icon select EC2
   2. On the left menu find > Network & Security > Key Pairs
      1. Click on the button “Create key pair
         1. Give it a name
         2. Keep RSA selected
         3. Keep .pem selected
         4. Add new tag
            1. Name and add the same name from 1.1
   3. After creating download the file and add top the secure repository

Instructions

Install Terraform and AWS CLI on local PC

- Windows / Mac OS / Linux
  - To install Terraform CLI, follow the Instruction here: https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli or https://phoenixnap.com/kb/how-to-install-terraform
  - To install AWS CLI, follow the Instruction here: https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html
  - To install GIT CLI, follow the Instruction here: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

Instructions

Running Terraform

1. Pull the repo with the scripts
    1. Clone the repo https://gitlab.com/terraform-assets-scripts/assets-scripts.git
    2. There are four files into the repo
        1. main.tf
        2. variables.tf
        3. install_docker.sh
        4. install_jenkins.sh
    3. Files explanation
        1. main.tf files holds the assets configuration that will be created, changed, or destroyed with some variables be flexible and reusable.
        2. variables.tf holds the assets information and configuration to be replaced on main.tf file
        3. install_docker.sh contain shell script commands as auxiliary to build the Docker EC2 asset.
        4. install_jenklins.sh contain shell script commands as auxiliary to build the Jenkins EC2 asset.
2. Start Terraform (Note: To run the terraform command you need to be in the same folder of the files)
    1. Open Shell Console or CMD for windows
    2. Run $ terraform init
3. Validate files structure
    1. Into the folder run $ terraform validate

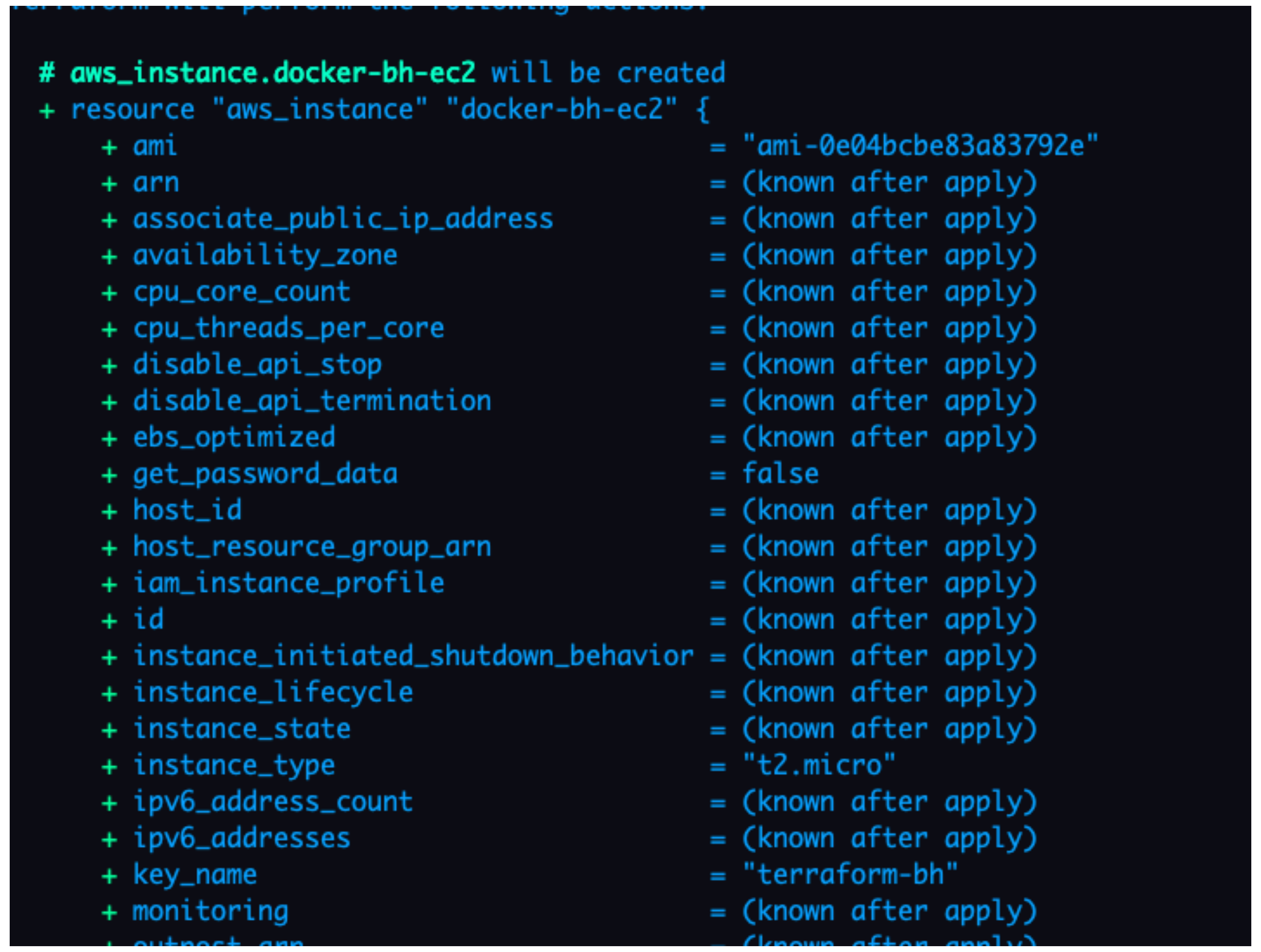
Instructions

Running Terraform

1. Plan the Infrastructure (Plan will generate an execution plan, showing you what actions will be taken without actually performing the planned actions.)

    1. Run $ terraform plan

        1. It will show something like this:



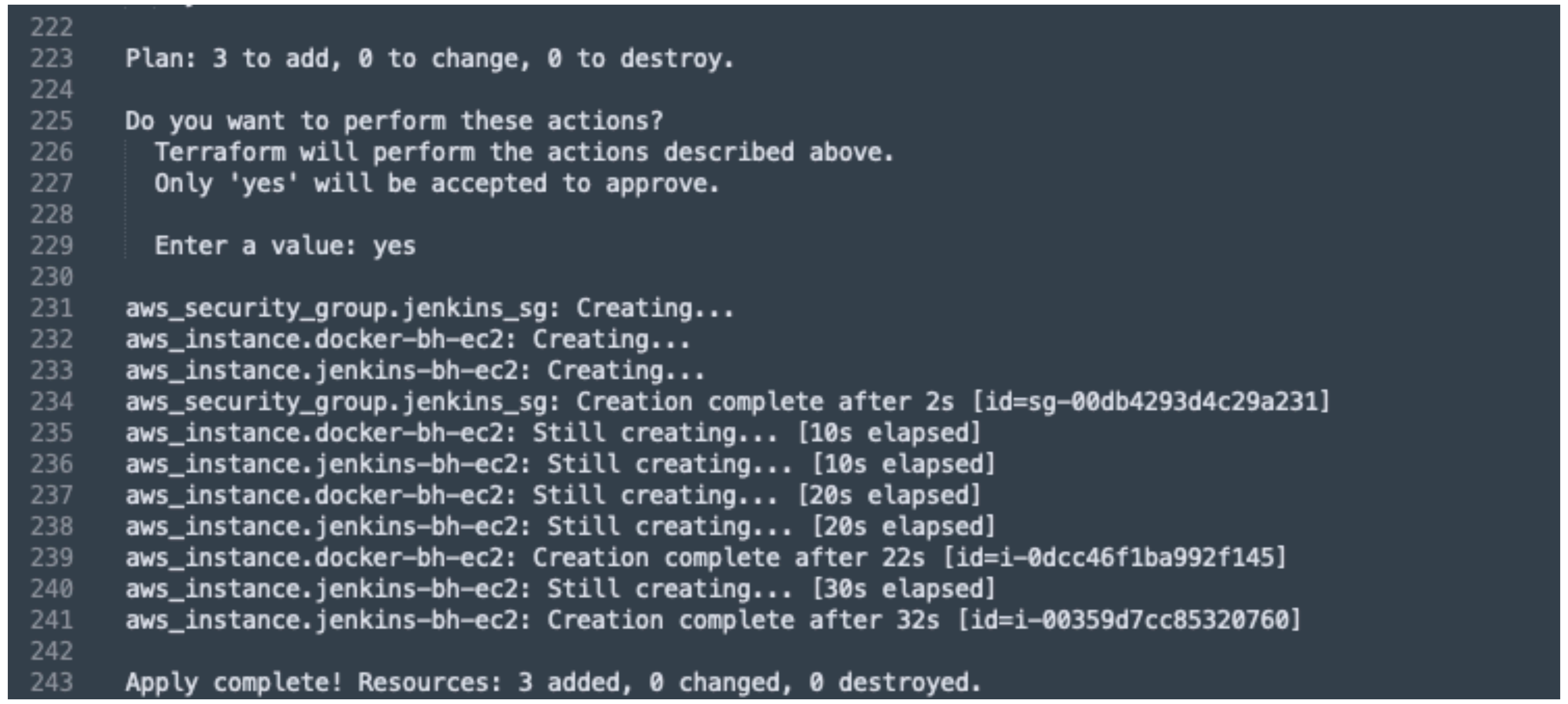
```
Terraform will perform the following actions:

  # aws_instance.docker-bh-ec2 will be created
  + resource "aws_instance" "docker-bh-ec2" {
      + ami                                  = "ami-0e04bcbe83a83792e"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = "terraform-bh"
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
```

    2. Analyse carefully the report and if everything is ok go to the next step.

Instructions

Running Terraform

1. Deploy Assets
    1. Apply (Create or update infrastructure depending on the configuration files. By default, a plan will be generated first and will need to be approved before it is applied)
        1. Run $ terraform apply
        2. At the end of the process it will show something like this

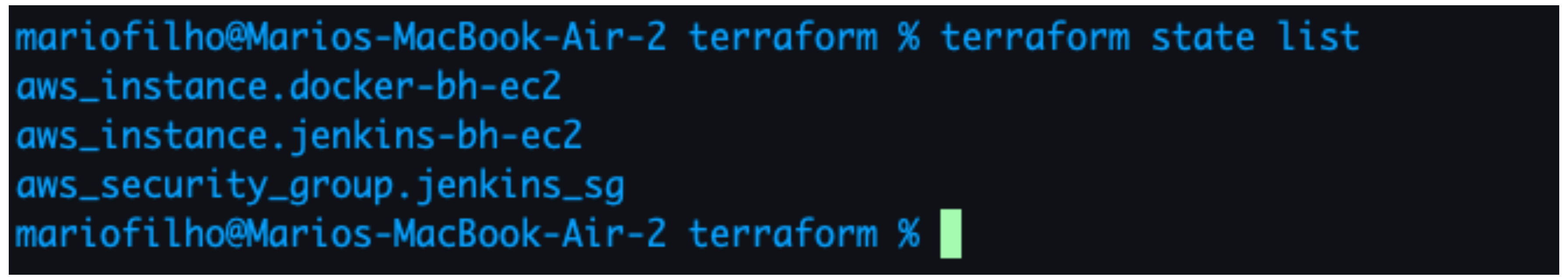


```
Plan: 3 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_security_group.jenkins_sg: Creating...
aws_instance.docker-bh-ec2: Creating...
aws_instance.jenkins-bh-ec2: Creating...
aws_security_group.jenkins_sg: Creation complete after 2s [id=sg-00db4293d4c29a231]
aws_instance.docker-bh-ec2: Still creating... [10s elapsed]
aws_instance.jenkins-bh-ec2: Still creating... [10s elapsed]
aws_instance.docker-bh-ec2: Still creating... [20s elapsed]
aws_instance.jenkins-bh-ec2: Still creating... [20s elapsed]
aws_instance.docker-bh-ec2: Creation complete after 22s [id=i-0dcc46f1ba992f145]
aws_instance.jenkins-bh-ec2: Still creating... [30s elapsed]
aws_instance.jenkins-bh-ec2: Creation complete after 32s [id=i-00359d7cc85320760]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
```

Instructions

Running Terraform

1. Check the assets deployed
    1. Check state (Lists out all the resources that are tracked in the current state file.)
        1. Run $ terraform state list
            1. It will show something like this



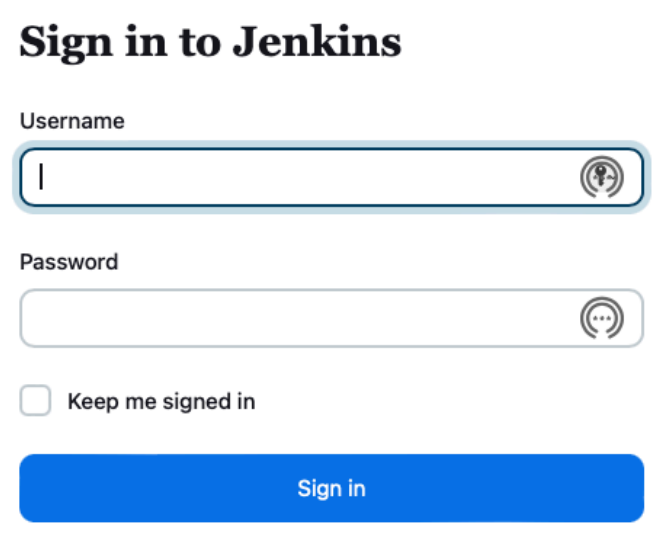
```
mariofilho@Marios-MacBook-Air-2 terraform % terraform state list
aws_instance.docker-bh-ec2
aws_instance.jenkins-bh-ec2
aws_security_group.jenkins_sg
mariofilho@Marios-MacBook-Air-2 terraform %
```

        2. Run $ terraform state pull > state.tfstate

Instructions

Set up Jenkins Pipeline

- Open the asset already deployed https://bh.jenkins/login?
  - Enter the user and password



Sign in to Jenkins

Username

Password

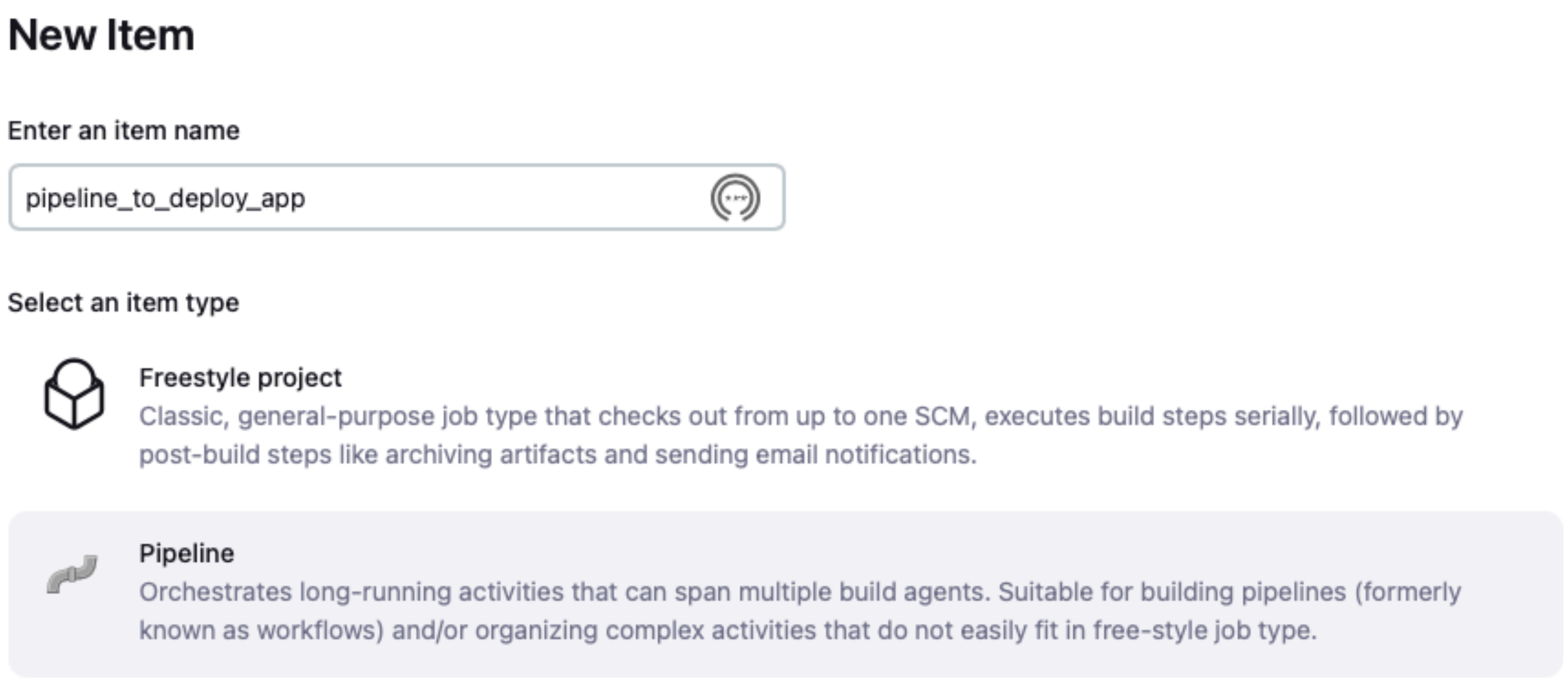
□ Keep me signed in

  - Click on + New Item

Instructions

Set up Jenkins Pipeline

- Add a name and select Pipeline



New Item

Enter an item name

pipeline_to_deploy_app

Select an item type

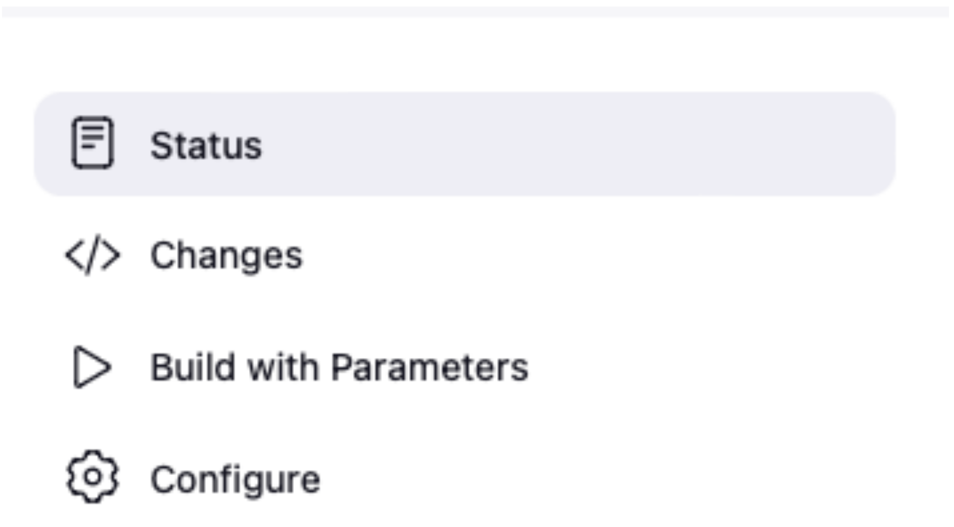
Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

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.

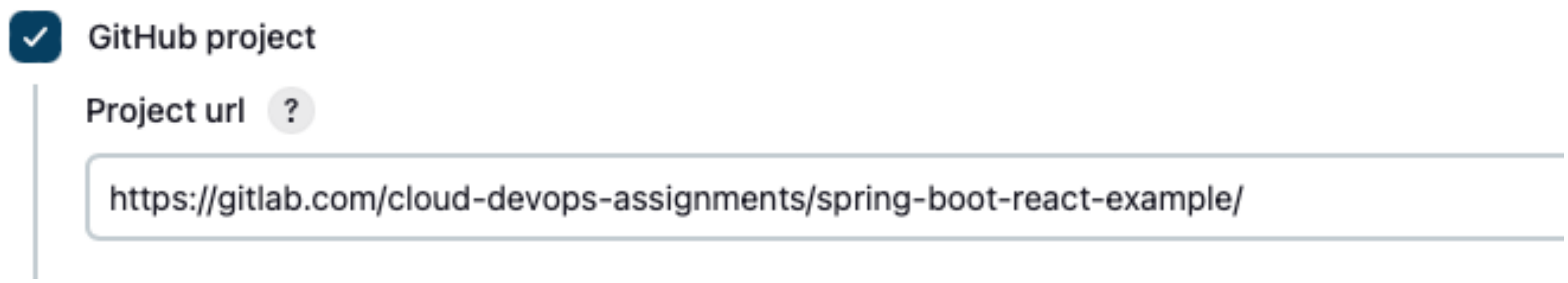
Instructions

Set up Jenkins Pipeline

- Click on Configure at the left panel



- Give a description, check GitHub project and add the url



☑ GitHub project

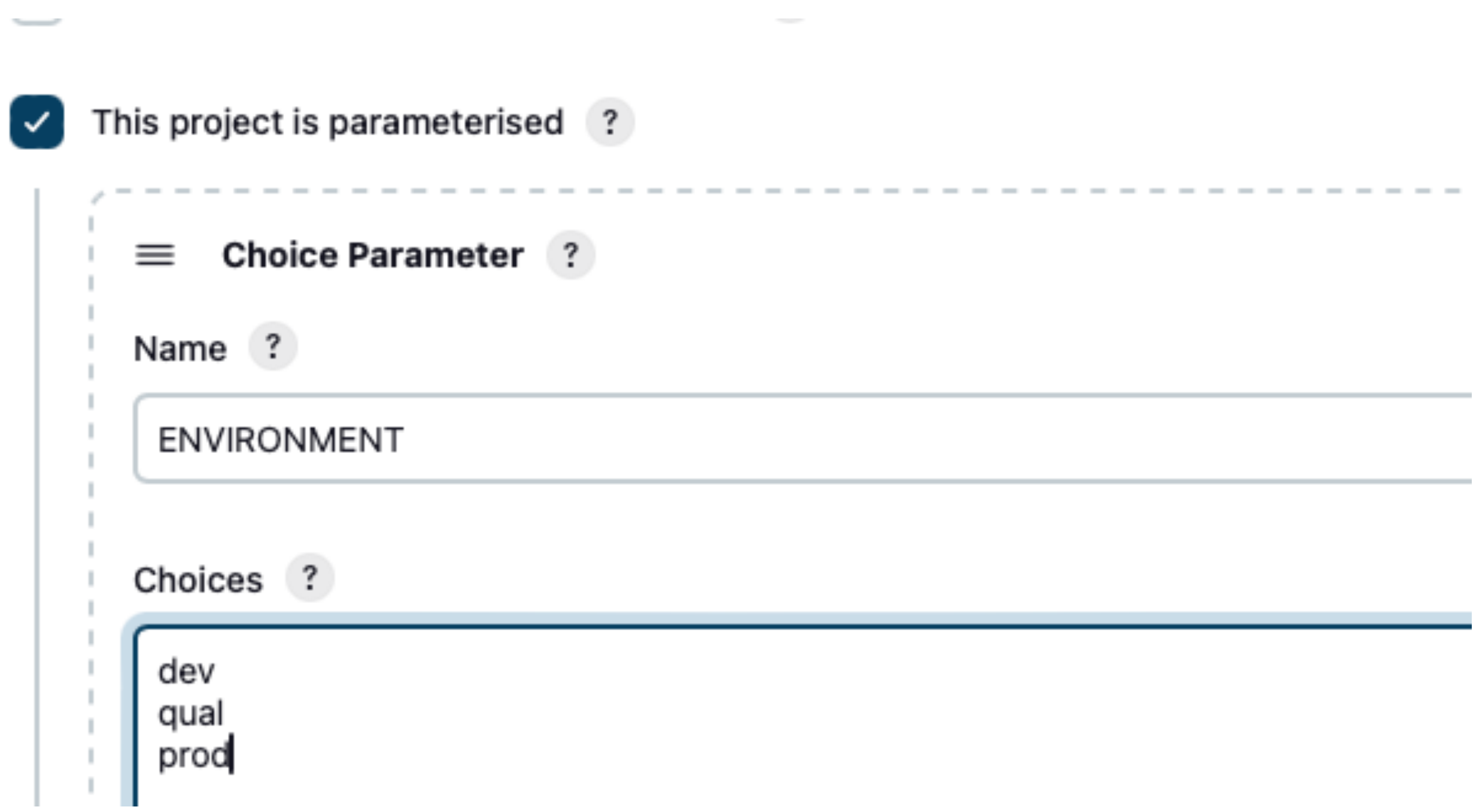
Project url ?

https://gitlab.com/cloud-devops-assignments/spring-boot-react-example/

Instructions

Set up Jenkins Pipeline

- Set a parameter to deploy across multiple environments



☑ This project is parameterised

Choice Parameter

Name: ENVIRONMENT

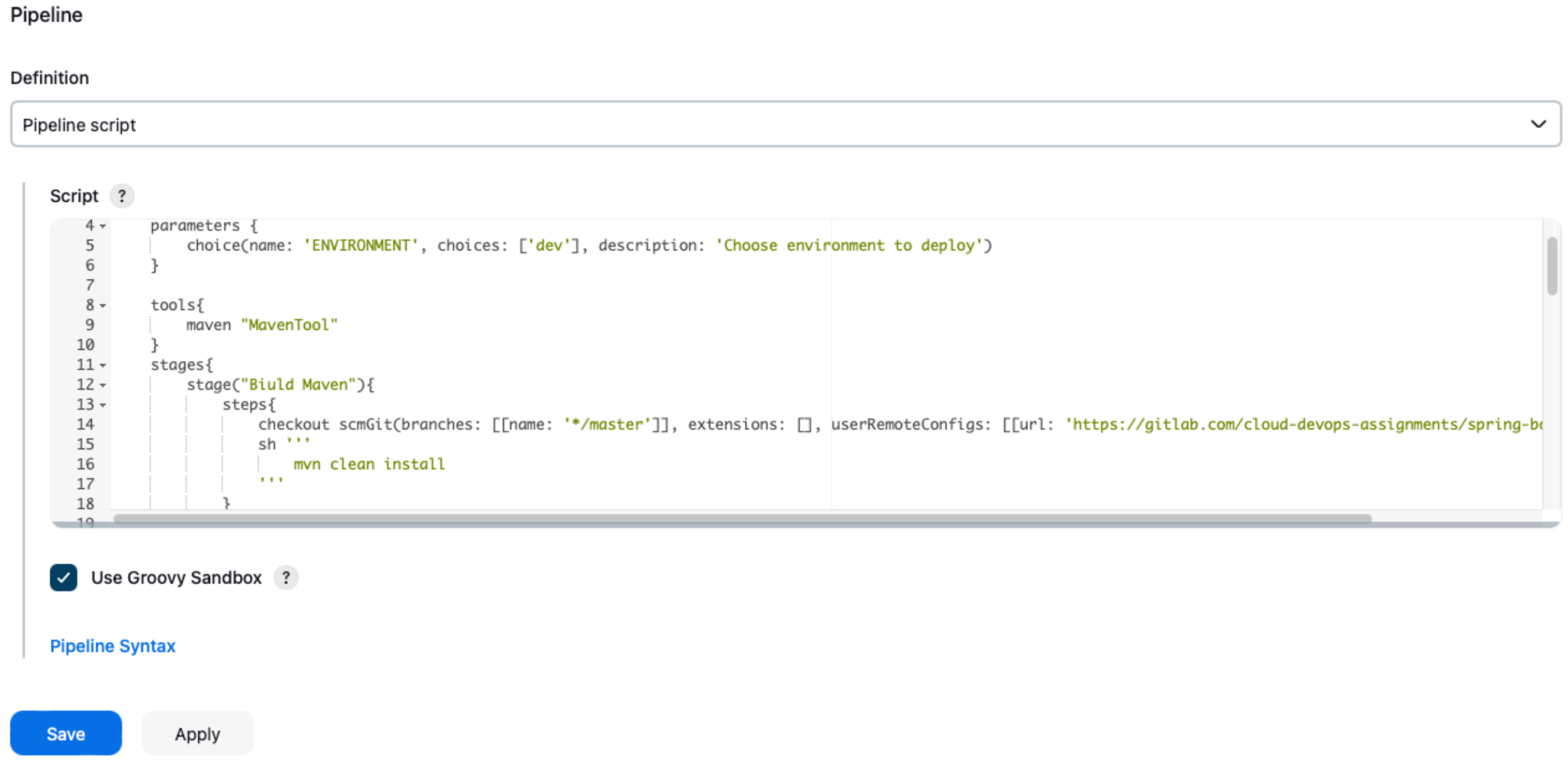
Choices:

```
dev
qual
prod
```

Instructions

Set up Jenkins Pipeline

- Paste the bh_app_pipeline content into the script and save it
-



Pipeline

Definition

Pipeline script

Script ?

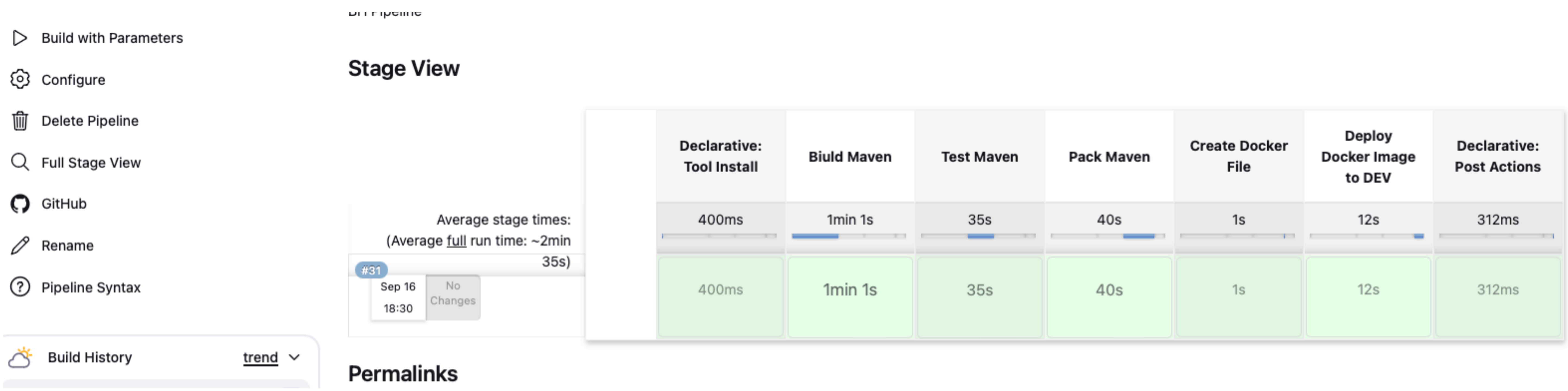
```
    parameters {
        choice(name: 'ENVIRONMENT', choices: ['dev'], description: 'Choose environment to deploy')
    }

    tools{
        maven "MavenTool"
    }
    stages{
        stage("Biuld Maven"){
            steps{
                checkout scmGit(branches: [[name: '*/master']], extensions: [], userRemoteConfigs: [[url: 'https://gitlab.com/cloud-devops-assignments/spring-b
                sh '''
                    mvn clean install
                '''
            }
```

☑ Use Groovy Sandbox ?

Instructions

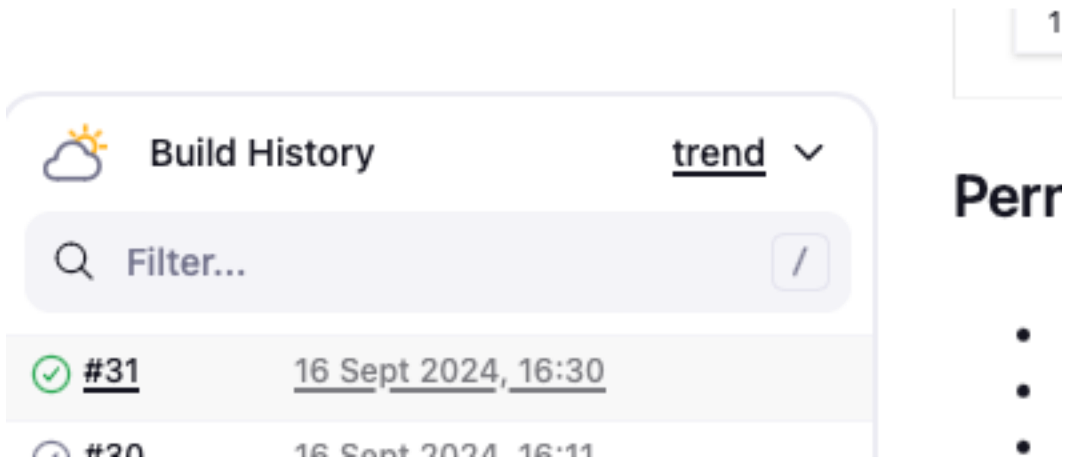
Executing the pipeline

- Click on “Build with Parameters” on the left panel and wait for the stage view



	Declarative: Tool Install	Biuld Maven	Test Maven	Pack Maven	Create Docker File	Deploy Docker Image to DEV	Declarative: Post Actions
Average stage times: (Average full run time: ~2min 35s)	400ms	1min 1s	35s	40s	1s	12s	312ms
#31 Sep 16 18:30 No Changes	400ms	1min 1s	35s	40s	1s	12s	312ms

- To check the console click oven the build number



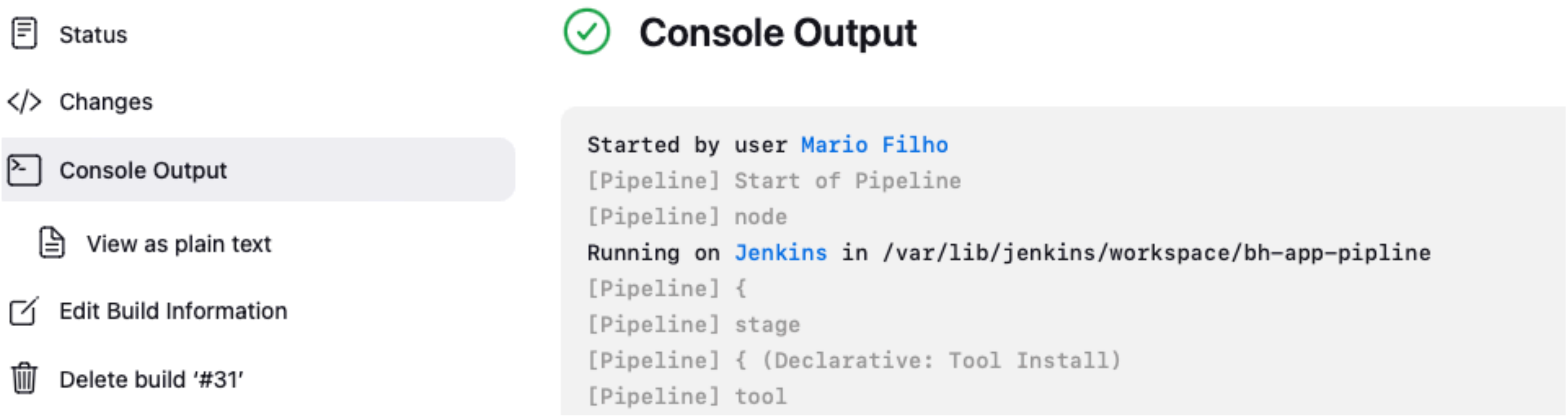
#31 16 Sept 2024, 16:30

#30 16 Sept 2024, 16:11

Instructions

Checking the results

- Click on Console Output



```
Started by user Mario Filho
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/bh-app-pipline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Tool Install)
[Pipeline] tool
```