EXPERIMENT - 2

Aim: To Build Your Application using AWS Code Build and Deploy on S3 / SEBS using AWS Code Pipeline, deploy Sample Application on EC2 instance using AWS Code Deploy.

Theory:

AWS Code Pipeline:

Continuous deployment allows you to deploy revisions to a production environment automatically without explicit approval from a developer, making the entire software release process automated. You will create the pipeline using AWS CodePipeline, a service that builds, tests, and deploys your code every time there is a code change. You will use your GitHub account, an Amazon Simple Storage Service (S3) bucket, or an AWS CodeCommit repository as the source location for the sample app's code. You will also use AWS Elastic Beanstalk as the deployment target for the sample app. Your completed pipeline will be able to detect changes made to the source repository containing the sample app and then automatically update your live sample app.

AWS Code Build:

AWS CodeBuild is a fully managed build service in the cloud. CodeBuild compiles your source code, runs unit tests, and produces artifacts that are ready to deploy. CodeBuild eliminates the need to provision, manage, and scale your own build servers. It provides prepackaged build environments for popular programming languages and build tools such as Apache Maven, Gradle, and more. You can also customize build environments in CodeBuild to use your own build tools. CodeBuild scales automatically to meet peak build requests.

You can add CodeBuild as a build or test action to the build or test stage of a pipeline in AWS CodePipeline. AWS CodePipeline is a continuous delivery service that you can use to model, visualize, and automate the steps required to release your code. This includes building your code. A pipeline is a workflow construct that describes how code changes go through a release process.In this Experiment, you use AWS CodeBuild to build a collection of sample source code input files (build input artifacts or build input) into a deployable version of the source code (build output artifact or build output). Specifically, you instruct CodeBuild to use Apache Maven, a common build tool, to build a set of Java class files into a Java Archive (JAR) file.

AWS S3:

Amazon S3 (Simple Storage Service) is a highly scalable, durable, and secure object storage service provided by AWS. It is designed to store and retrieve any amount of data from anywhere on the web. Here's an overview of some key concepts and features of AWS S3

Steps:

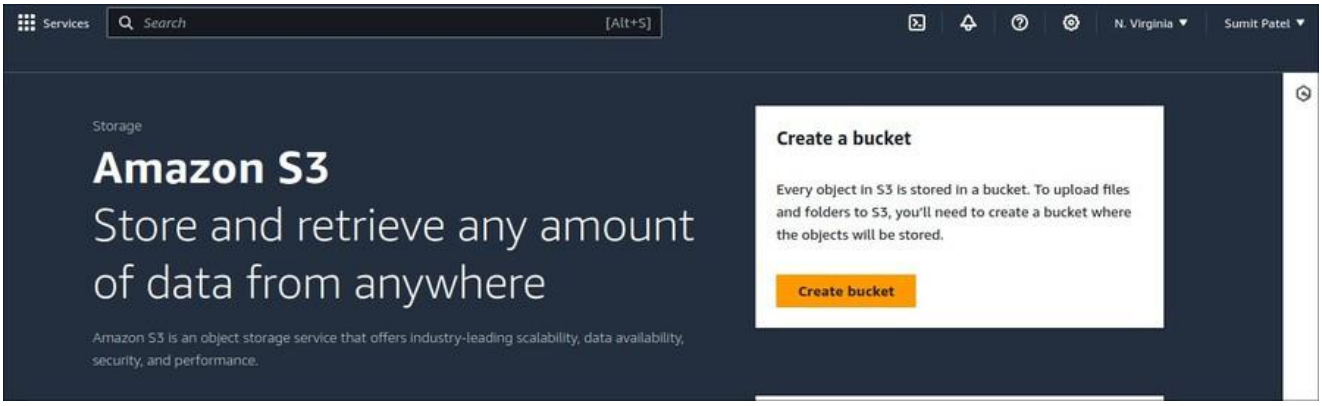
1. Create your project / source code. / Get your any project.

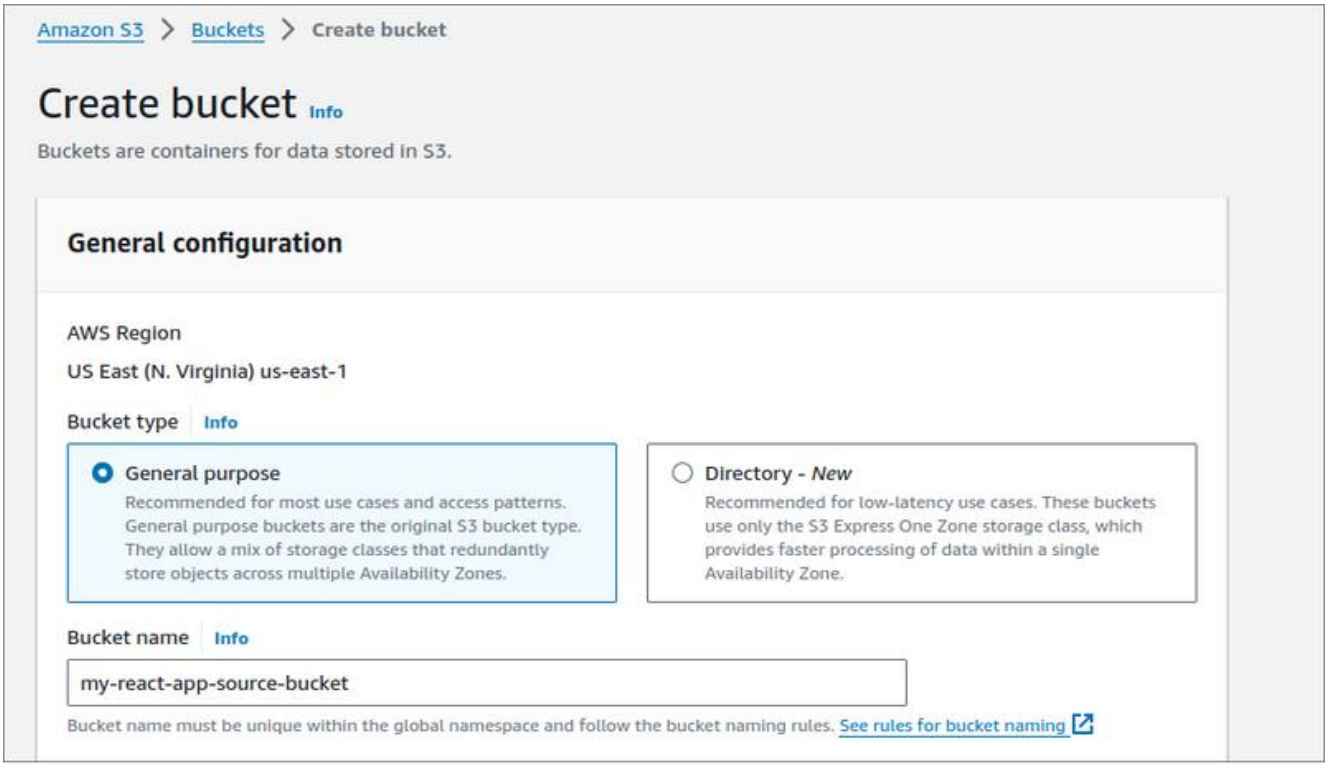
In this step you can need to choose any existing project you wanted to deploy, or create new project / source code. You can push to any git server.

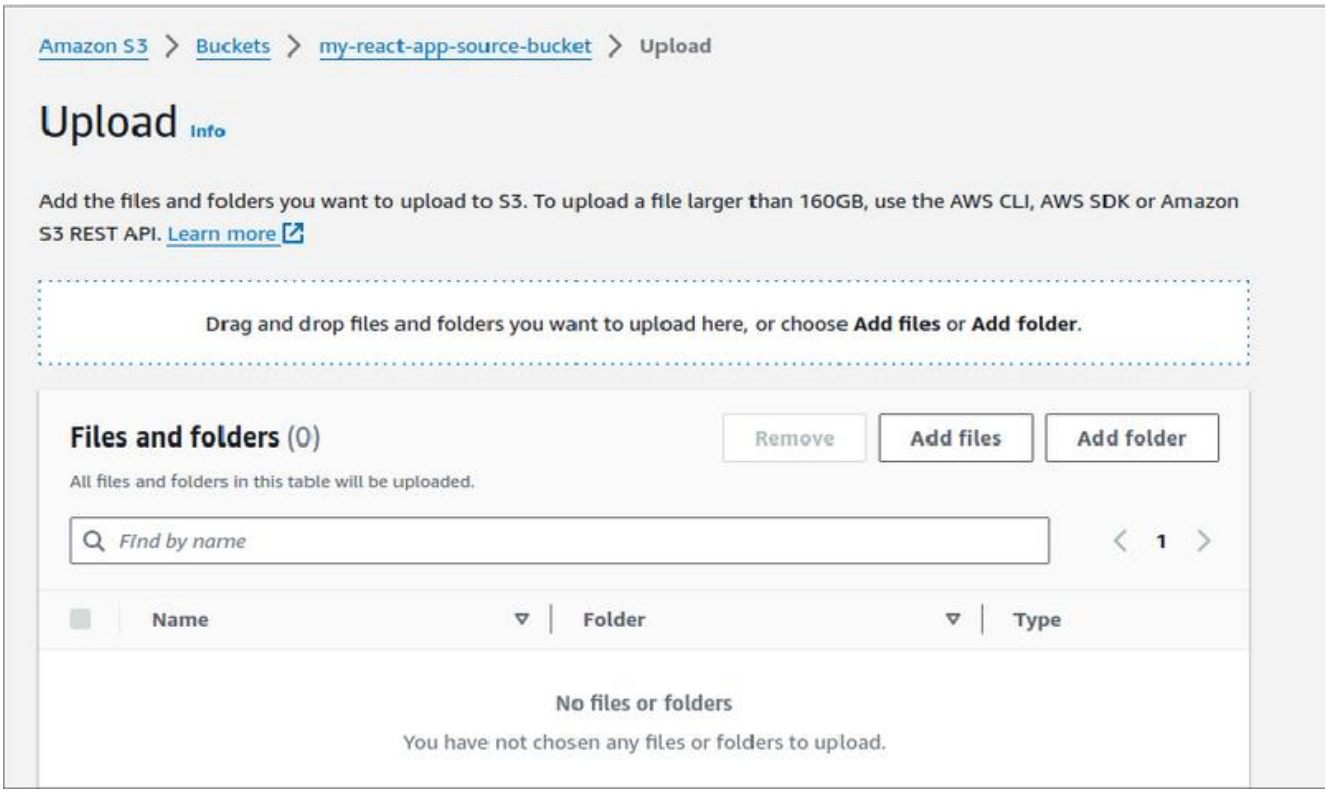
2. Create a buildspec.yml file

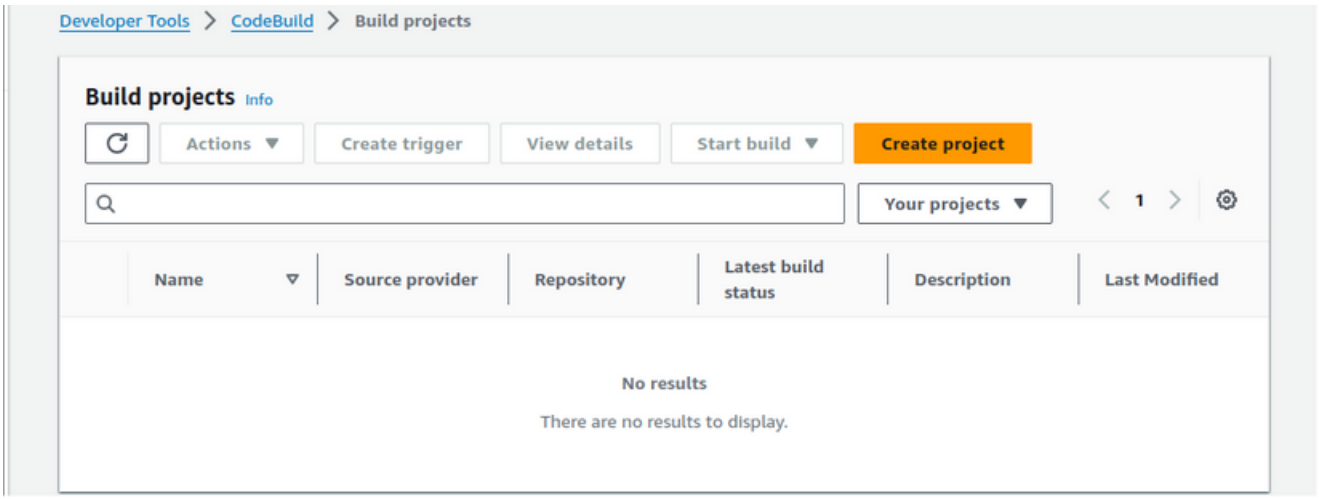
In this step, you create a build specification (build spec) file. A buildspec is a collection of build commands and related settings, in YAML format, that CodeBuild uses to run a build. Without a build spec, CodeBuild cannot successfully convert your build input into build output or locate the build output artifact in the build environment to upload to your output bucket. Create this file, name it buildspec.yml, and then save it in the root (top level) directory.

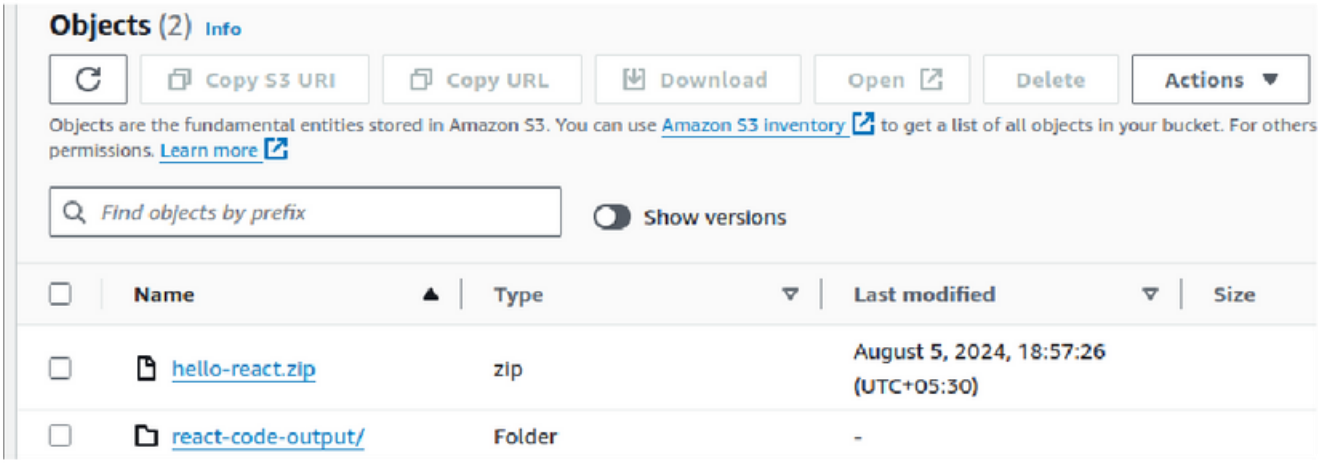
3.

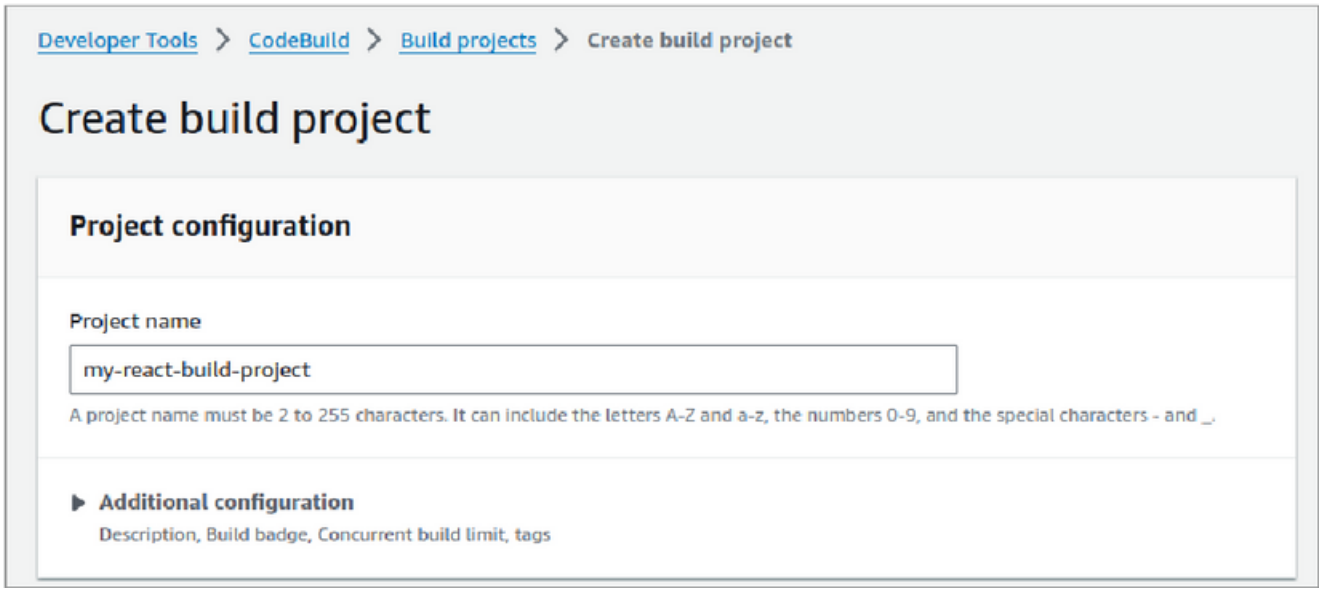


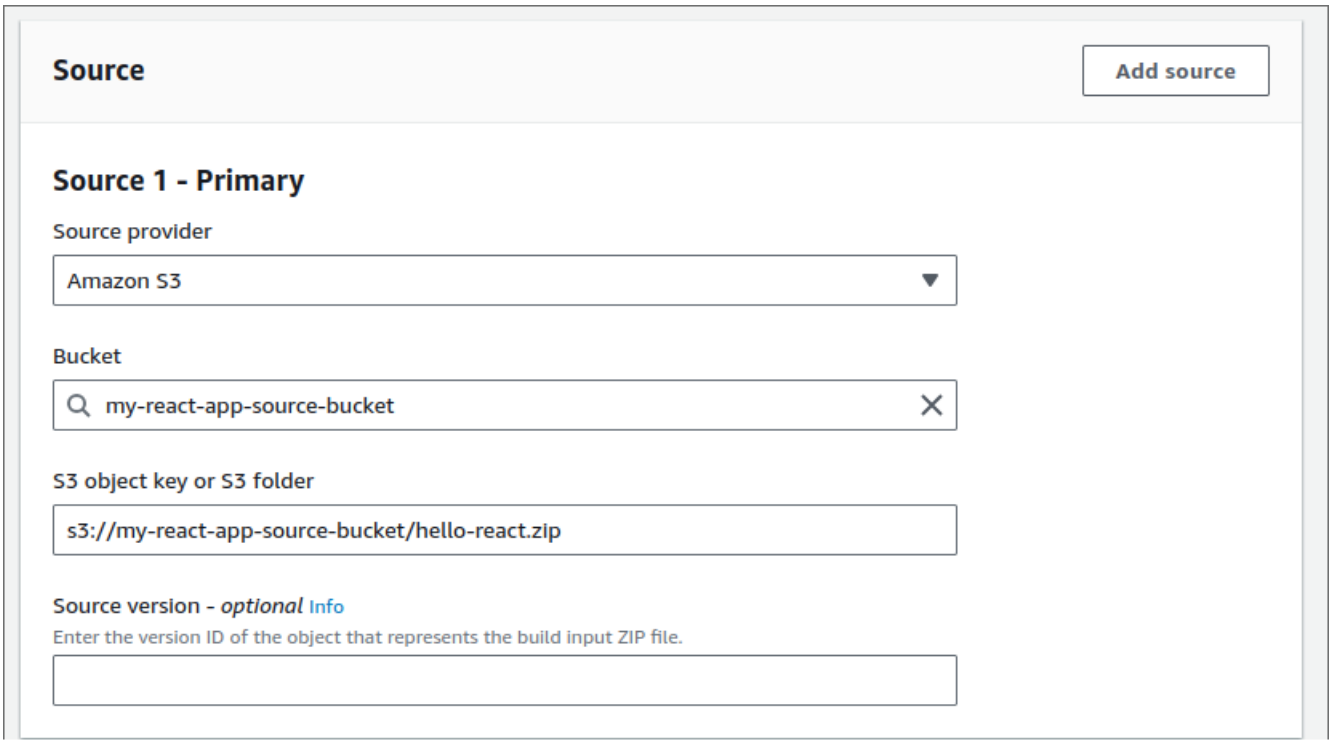

Amazon S3 > Buckets > Create bucket
Create bucket Info
Buckets are containers for data stored in S3.
General configuration
AWS Region
US East (N. Virginia) us-east-1
Bucket type Info
General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.
Directory - New
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.
Bucket name Info
my-react-app-source-bucket
Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming

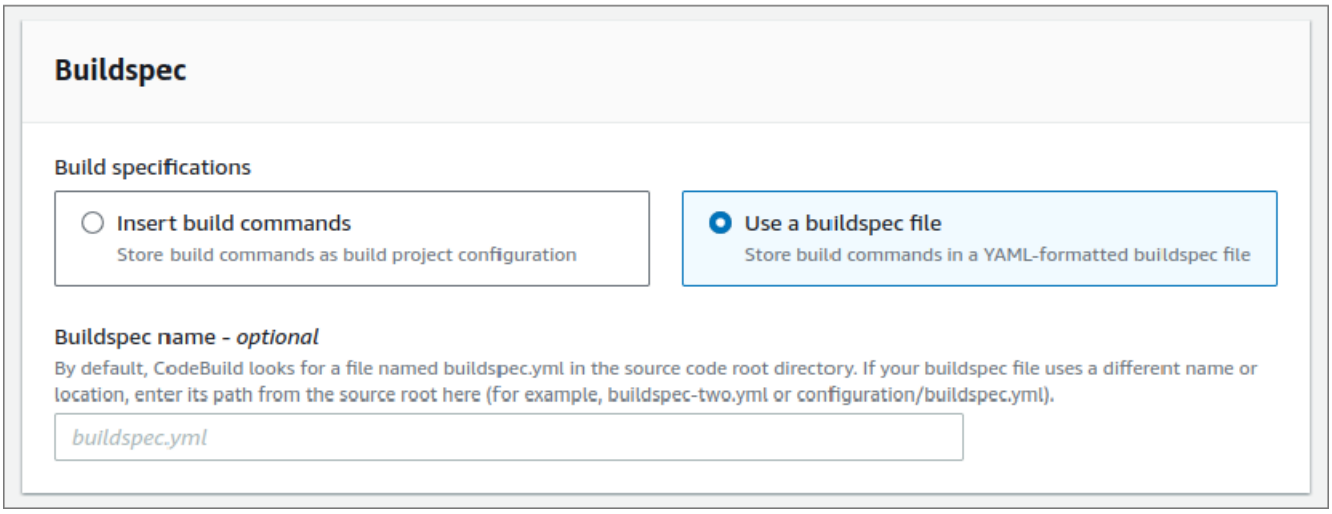

Amazon S3 > Buckets > my-react-app-source-bucket > Upload
Upload Info
Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. Learn more
Drag and drop files and folders you want to upload here, or choose Add files or Add folder.
Files and folders (0)
Remove
Add files
Add folder
All files and folders in this table will be uploaded.
Find by name
1
Name
Folder
Type
No files or folders
You have not chosen any files or folders to upload.

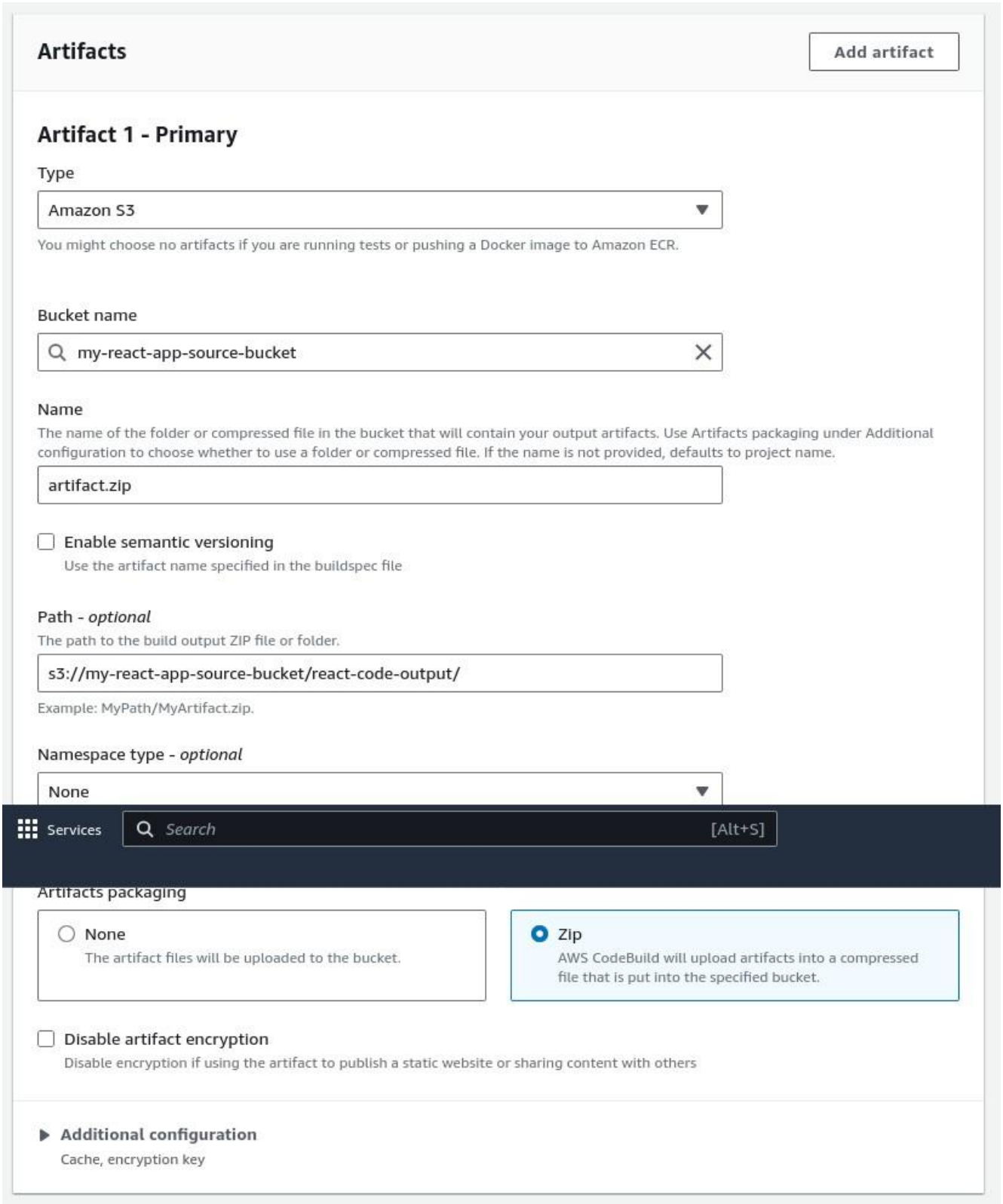

Developer Tools > CodeBuild > Build projects
Build projects Info
Actions
Create trigger
View details
Start build
Create project
Your projects
1
Name
Source provider
Repository
Latest build status
Description
Last Modified
No results
There are no results to display.

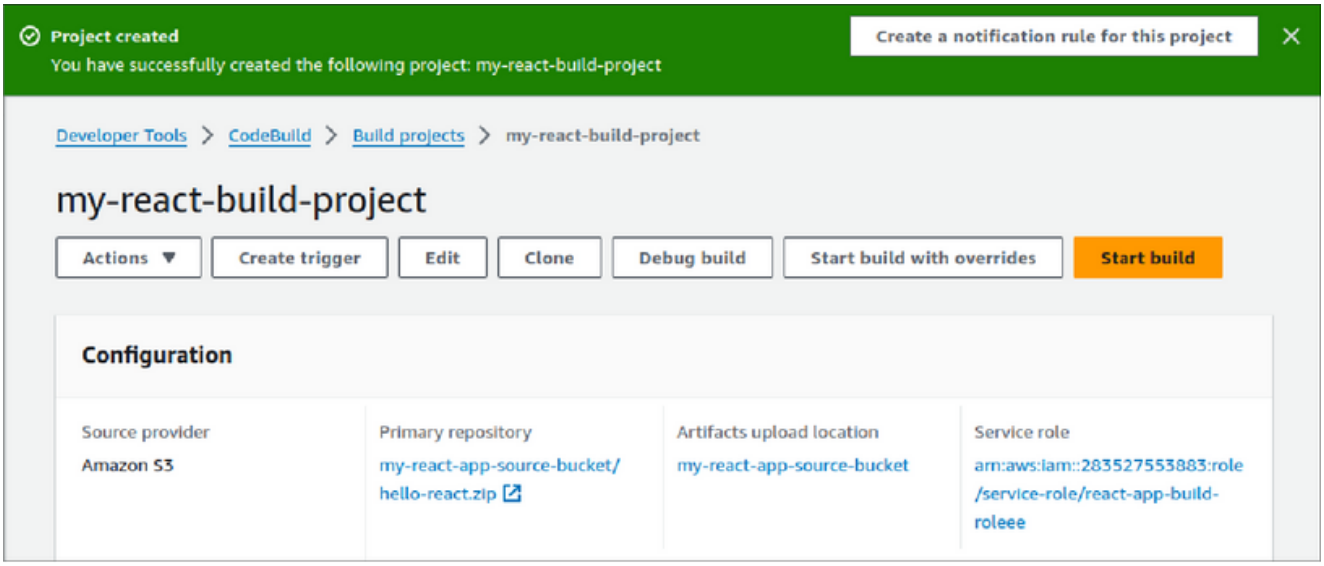

Objects (2) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others permissions. Learn more
Find objects by prefix
Show versions
Name
Type
Last modified
Size
hello-react.zip
zip
August 5, 2024, 18:57:26 (UTC+05:30)
react-code-output/
Folder
-

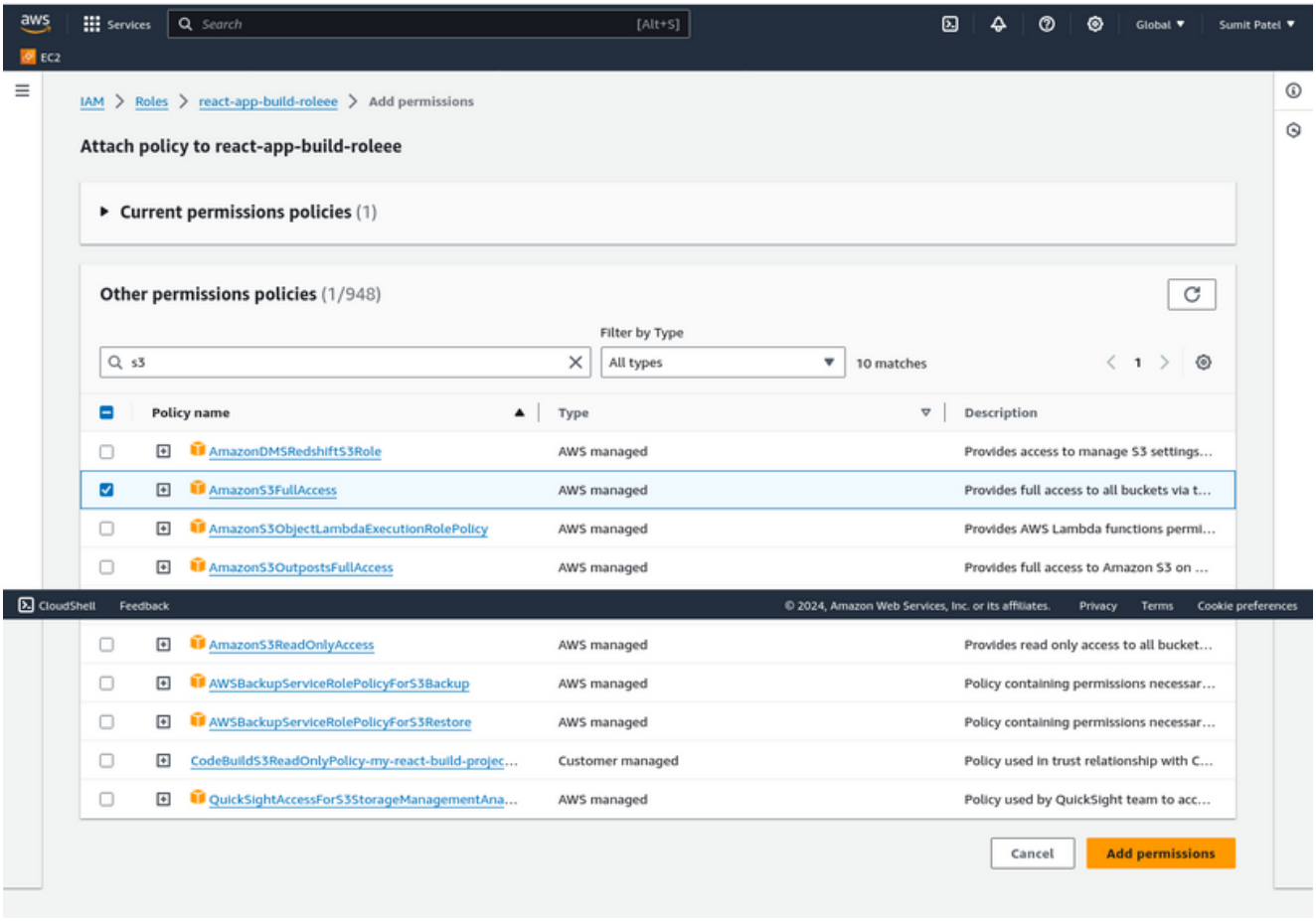

Developer Tools > CodeBuild > Build projects > Create build project
Create build project
Project configuration
Project name
my-react-build-project
A project name must be 2 to 255 characters. It can include the letters A-Z and a-z, the numbers 0-9, and the special characters - and _.
Additional configuration
Description, Build badge, Concurrent build limit, tags

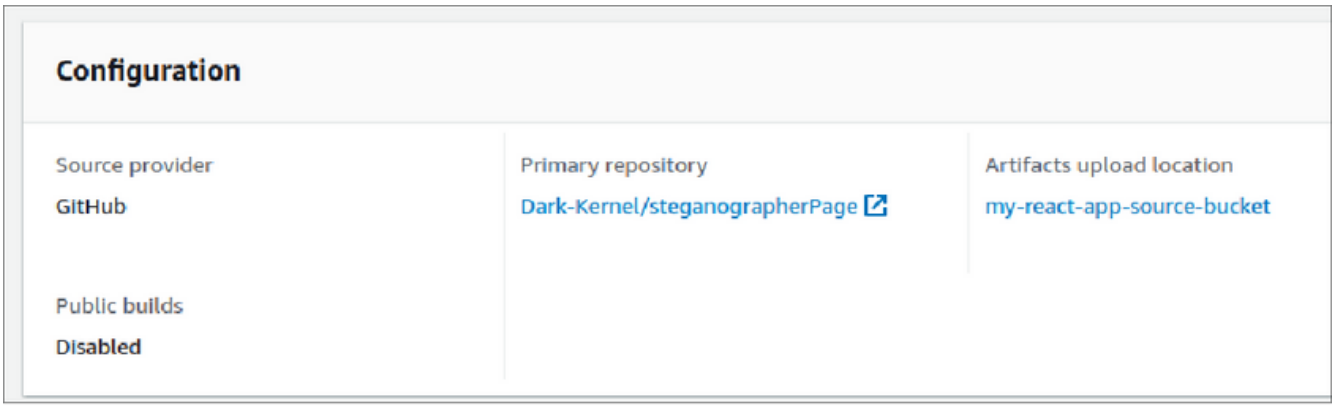

Source
Add source
Source 1 - Primary
Source provider
Amazon S3
Bucket
my-react-app-source-bucket
S3 object key or S3 folder
s3://my-react-app-source-bucket/hello-react.zip
Source version - optional Info
Enter the version ID of the object that represents the build input ZIP file.

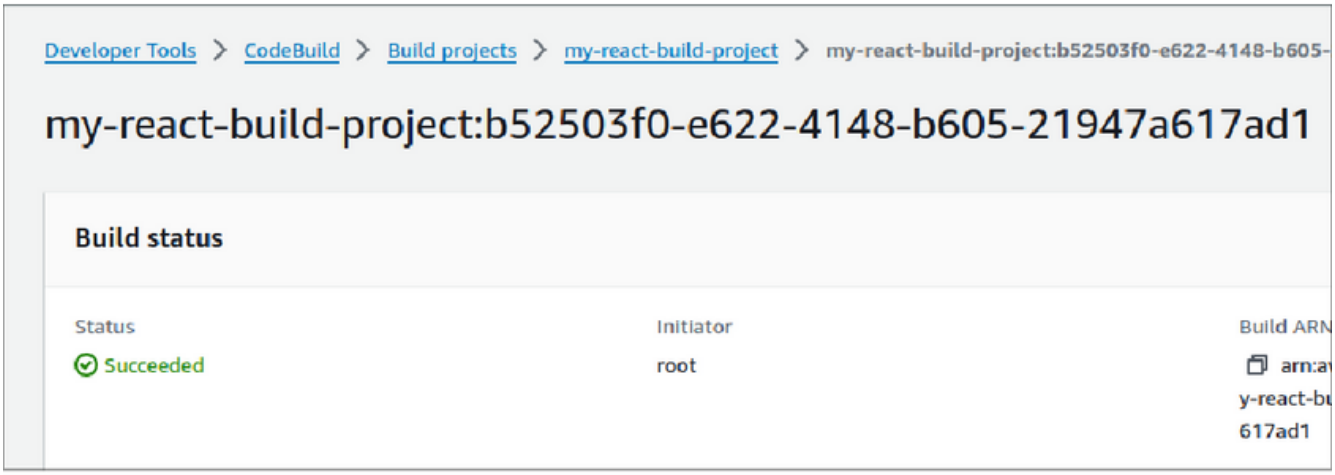

Buildspec
Build specifications
Insert build commands
Store build commands as build project configuration
Use a buildspec file
Store build commands in a YAML-formatted buildspec file
Buildspec name - optional
By default, CodeBuild looks for a file named buildspec.yml in the source code root directory. If your buildspec file uses a different name or location, enter its path from the source root here (for example, buildspec-two.yml or configuration/buildspec.yml).
buildspec.yml

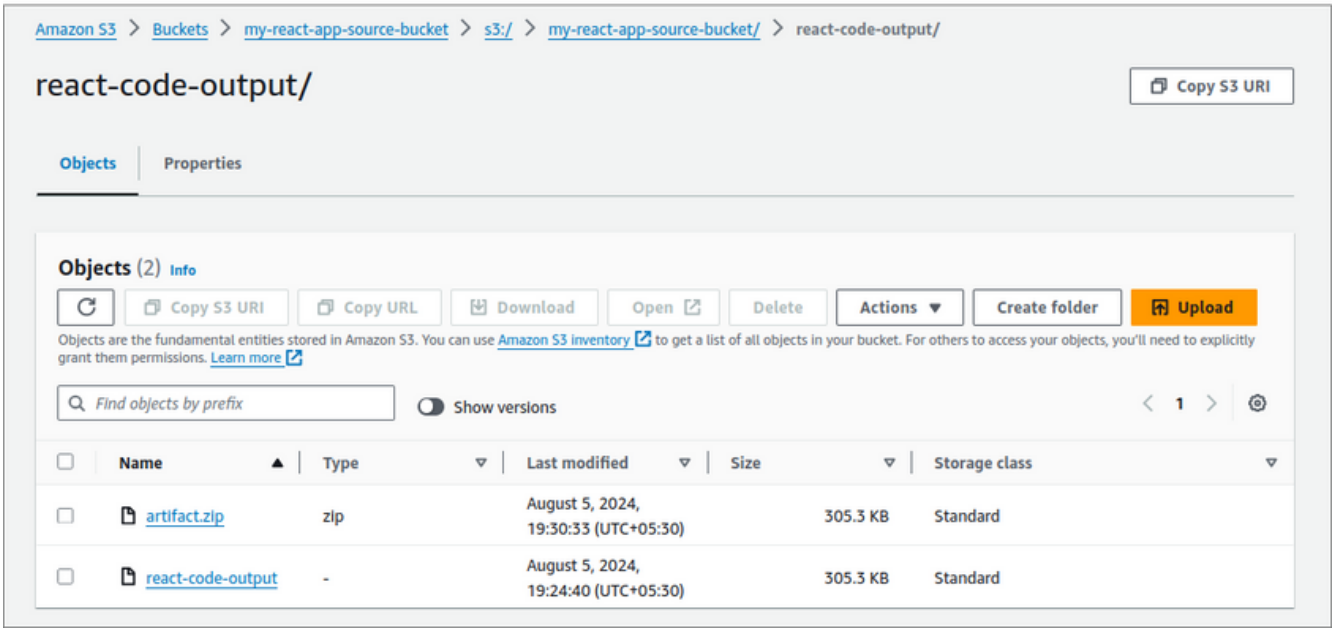

Artifacts
Add artifact
Artifact 1 - Primary
Type
Amazon S3
You might choose no artifacts if you are running tests or pushing a Docker image to Amazon ECR.
Bucket name
my-react-app-source-bucket
Name
The name of the folder or compressed file in the bucket that will contain your output artifacts. Use Artifacts packaging under Additional configuration to choose whether to use a folder or compressed file. If the name is not provided, defaults to project name.
artifact.zip
Enable semantic versioning
Use the artifact name specified in the buildspec file
Path - optional
The path to the build output ZIP file or folder.
s3://my-react-app-source-bucket/react-code-output/
Example: MyPath/MyArtifact.zip.
Namespace type - optional
None
Services
Search
[Alt+S]
Artifacts packaging
None
The artifact files will be uploaded to the bucket.
Zip
AWS CodeBuild will upload artifacts into a compressed file that is put into the specified bucket.
Disable artifact encryption
Disable encryption if using the artifact to publish a static website or sharing content with others
Additional configuration
Cache, encryption key

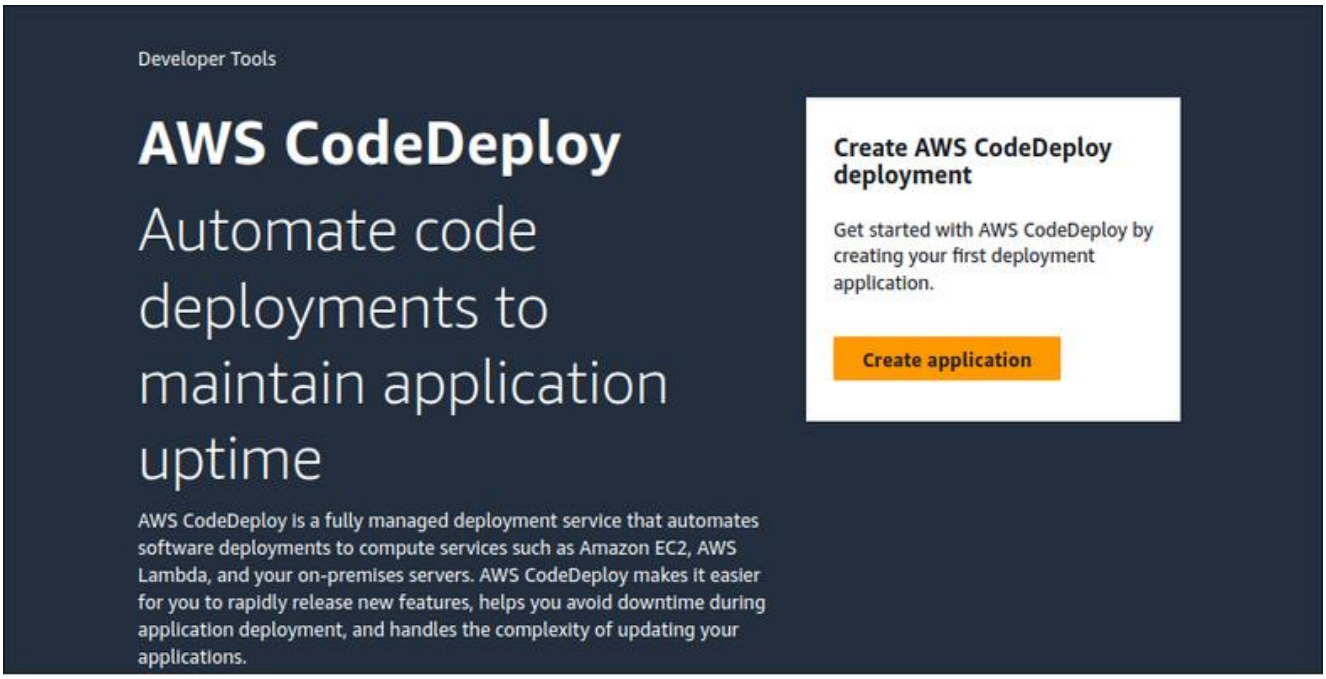

Project created
You have successfully created the following project: my-react-build-project
Create a notification rule for this project
Developer Tools > CodeBuild > Build projects > my-react-build-project
my-react-build-project
Actions
Create trigger
Edit
Clone
Debug build
Start build with overrides
Start build
Configuration
Source provider
Amazon S3
Primary repository
my-react-app-source-bucket/
hello-react.zip
Artifacts upload location
my-react-app-source-bucket
Service role
arn:aws:iam::283527553883:role
/service-role/react-app-build-
roleee

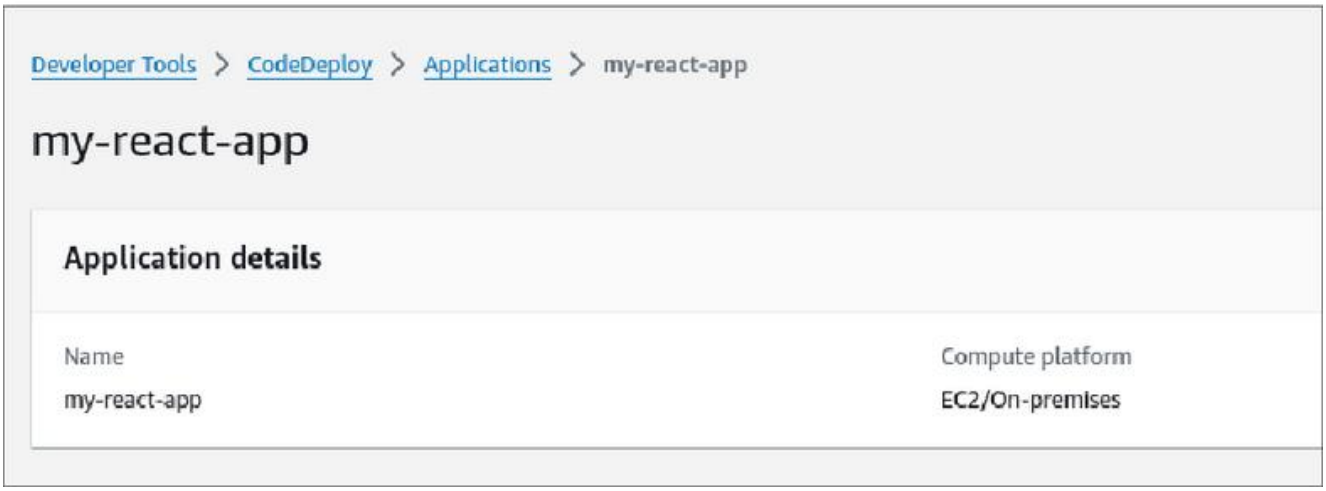

aws
Services
Search
[Alt+S]
Global
Sumit Patel
EC2
IAM > Roles > react-app-build-roleee > Add permissions
Attach policy to react-app-build-roleee
Current permissions policies (1)
Other permissions policies (1/948)
Filter by Type
s3
All types
10 matches
Policy name
Type
Description
AmazonDMSRedshiftS3Role
AWS managed
Provides access to manage S3 settings...
AmazonS3FullAccess
AWS managed
Provides full access to all buckets via t...
AmazonS3ObjectLambdaExecutionRolePolicy
AWS managed
Provides AWS Lambda functions permi...
AmazonS3OutpostsFullAccess
AWS managed
Provides full access to Amazon S3 on ...
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
AmazonS3ReadOnlyAccess
AWS managed
Provides read only access to all bucket...
AWSBackupServiceRolePolicyForS3Backup
AWS managed
Policy containing permissions necessar...
AWSBackupServiceRolePolicyForS3Restore
AWS managed
Policy containing permissions necessar...
CodeBuildS3ReadOnlyPolicy-my-react-build-projec...
Customer managed
Policy used in trust relationship with C...
QuickSightAccessForS3StorageManagementAna...
AWS managed
Policy used by QuickSight team to acc...
Cancel
Add permissions

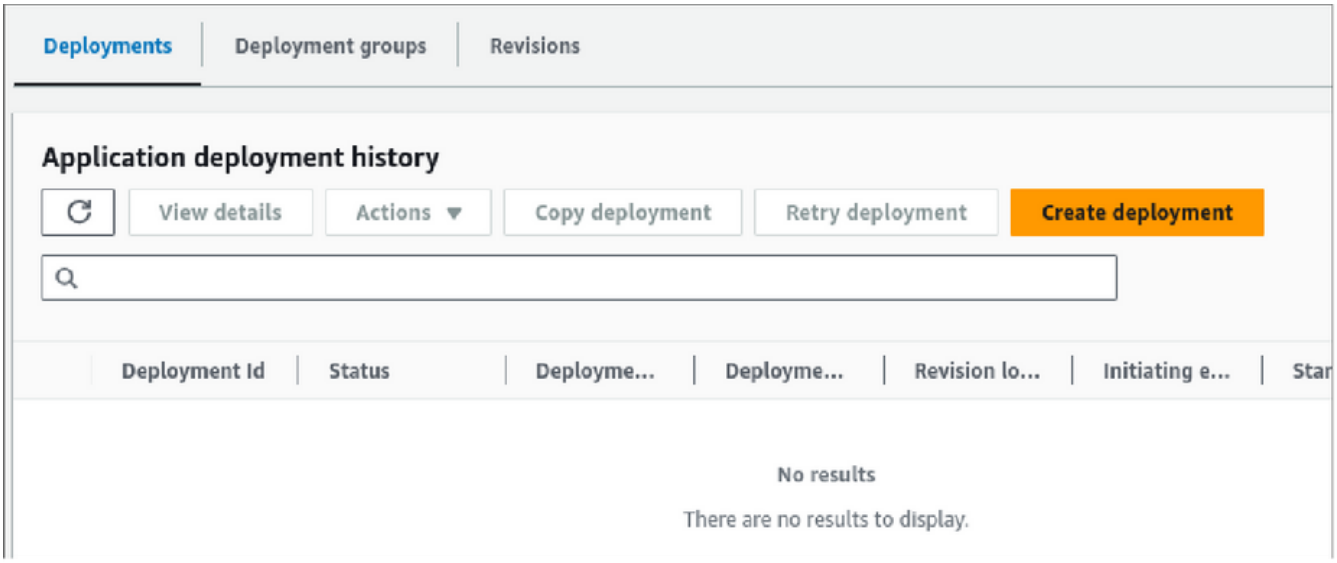

Configuration
Source provider
GitHub
Primary repository
Dark-Kernel/steganographerPage
Artifacts upload location
my-react-app-source-bucket
Public builds
Disabled

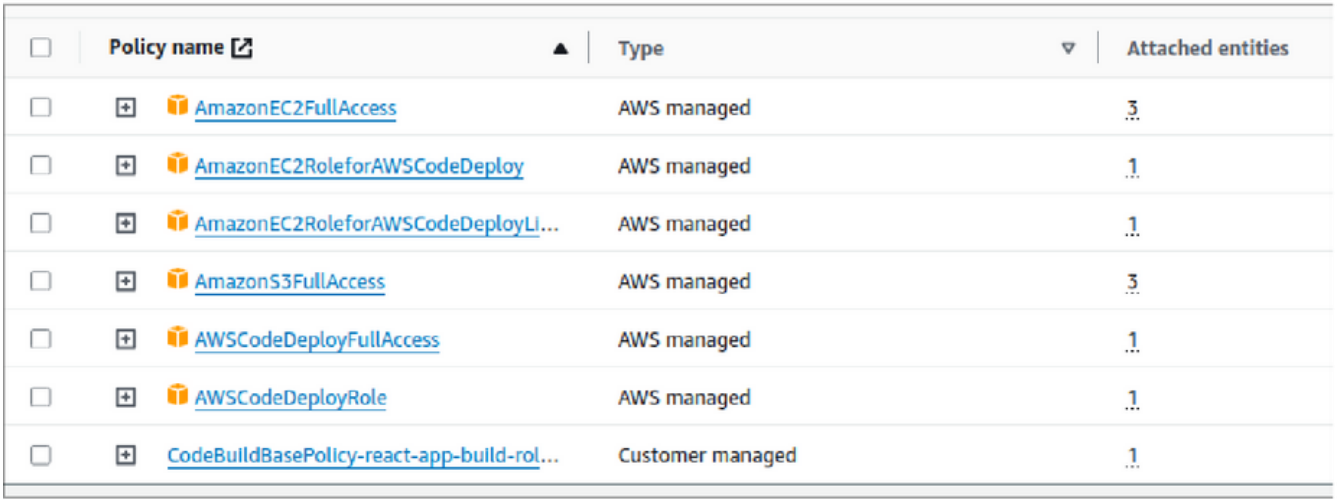

Developer Tools > CodeBuild > Build projects > my-react-build-project > my-react-build-project:b52503f0-e622-4148-b605-
my-react-build-project:b52503f0-e622-4148-b605-21947a617ad1
Build status
Status
Succeeded
Initiator
root
Build ARN

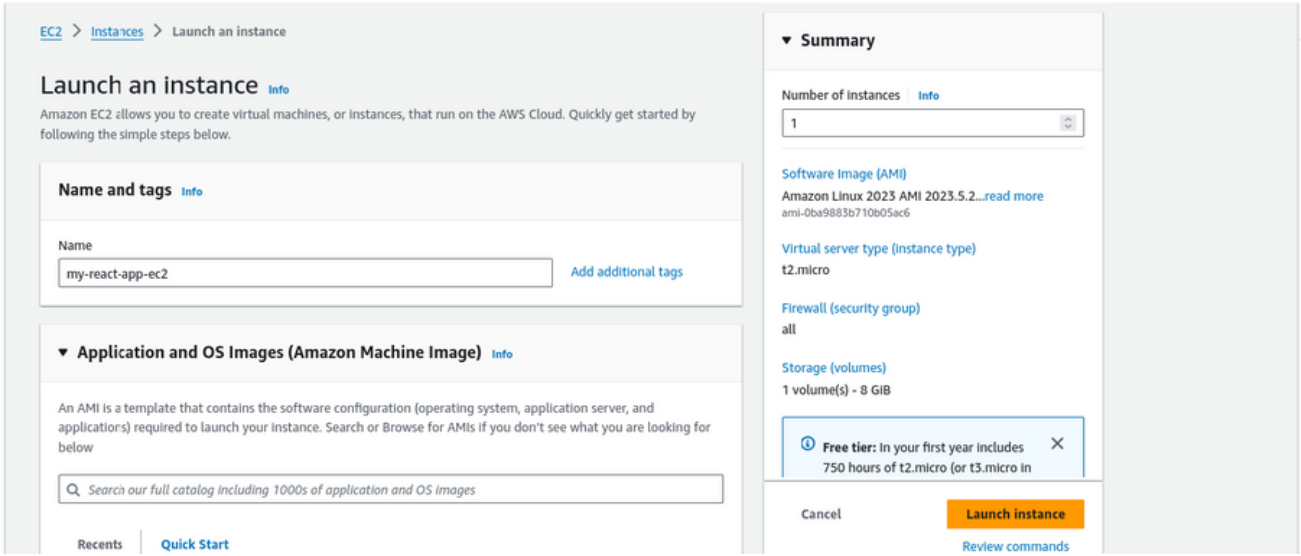

Amazon S3 > Buckets > my-react-app-source-bucket > s3:/ > my-react-app-source-bucket/ > react-code-output/
react-code-output/
Copy S3 URI
Objects
Properties
Objects (2) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
Show versions
1
Name
Type
Last modified
Size
Storage class
artifact.zip
zip
August 5, 2024, 19:30:33 (UTC+05:30)
305.3 KB
Standard
react-code-output
-
August 5, 2024, 19:24:40 (UTC+05:30)
305.3 KB
Standard

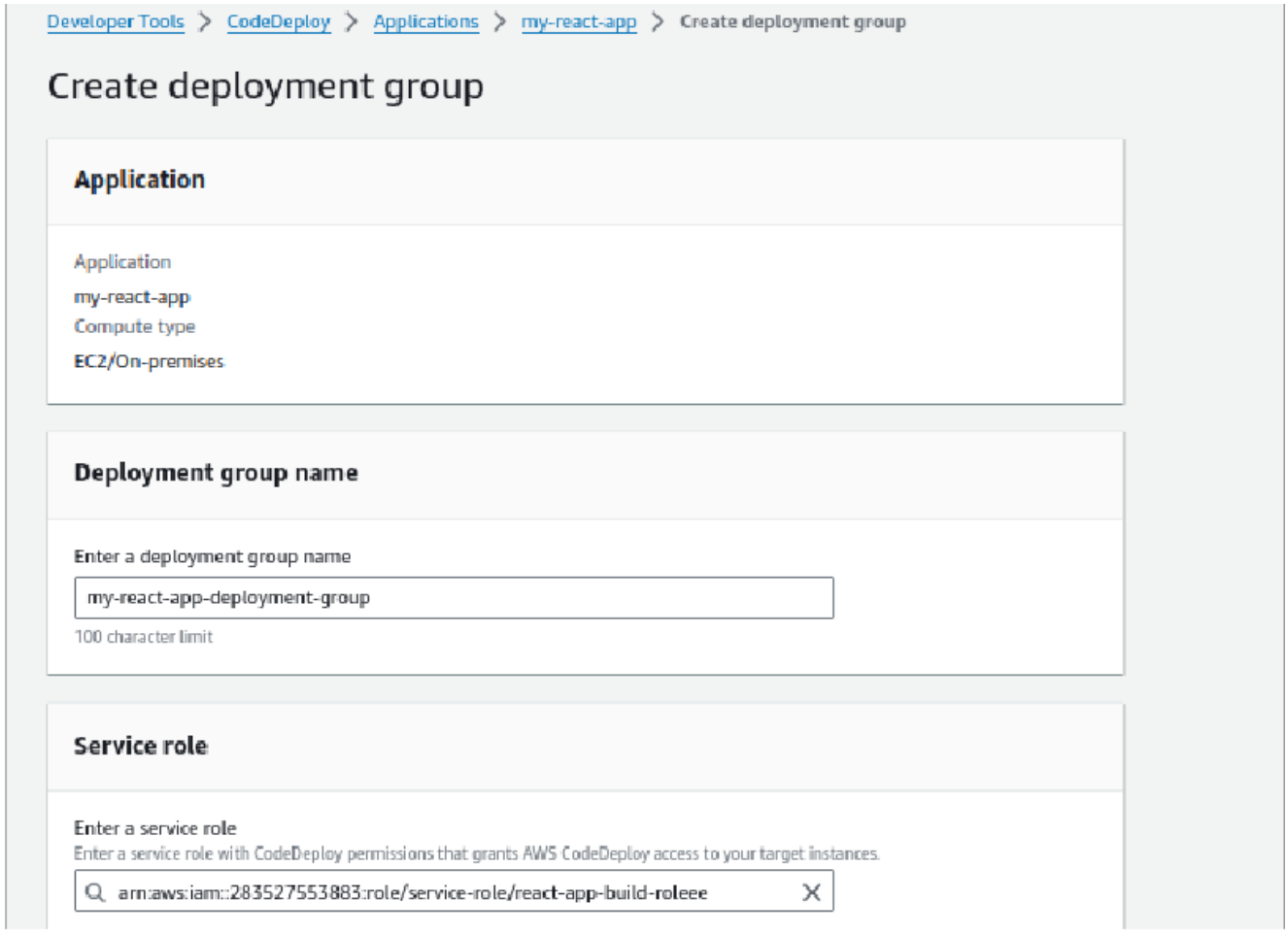

Developer Tools
AWS CodeDeploy
Automate code deployments to maintain application uptime
AWS CodeDeploy is a fully managed deployment service that automates software deployments to compute services such as Amazon EC2, AWS Lambda, and your on-premises servers. AWS CodeDeploy makes it easier for you to rapidly release new features, helps you avoid downtime during application deployment, and handles the complexity of updating your applications.
Create AWS CodeDeploy deployment
Get started with AWS CodeDeploy by creating your first deployment application.
Create application

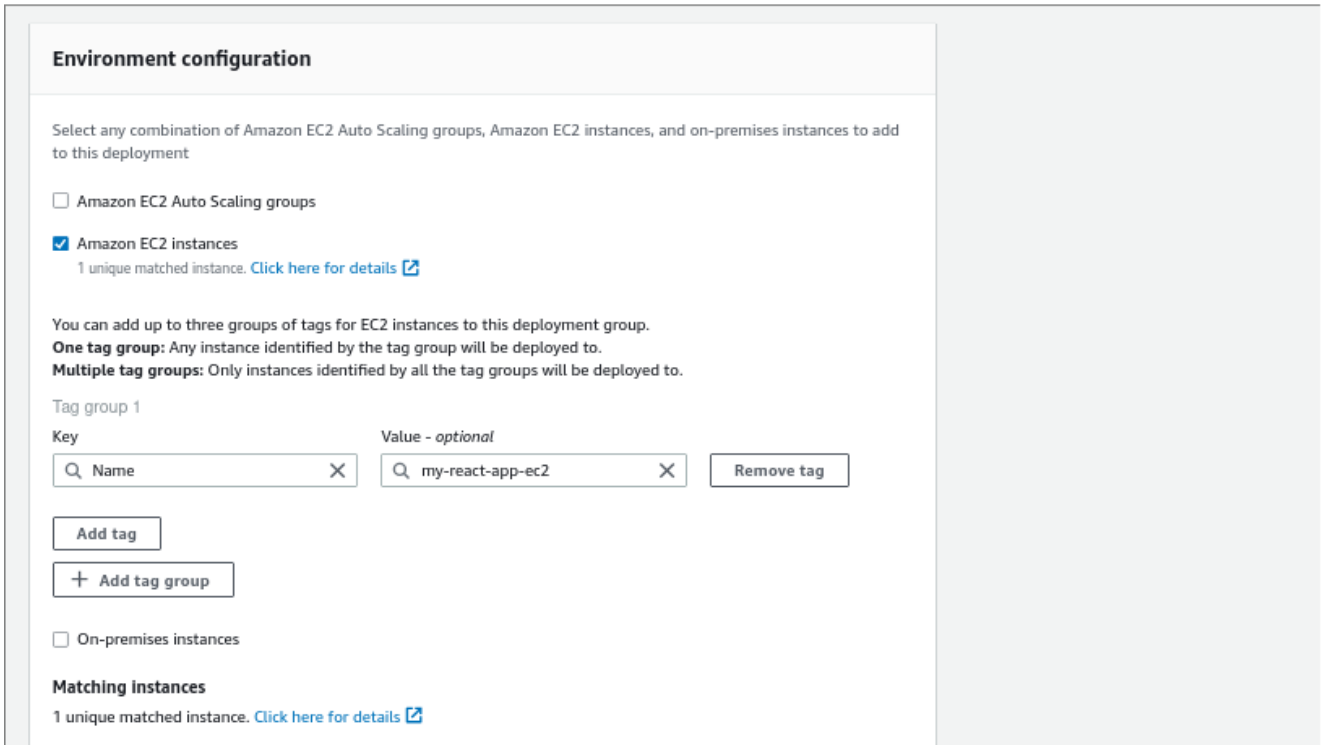

Developer Tools > CodeDeploy > Applications > my-react-app
my-react-app
Application details
Name
my-react-app
Compute platform
EC2/On-premises


Deployments
Deployment groups
Revisions
Application deployment history
View details
Actions
Copy deployment
Retry deployment
Create deployment
Deployment Id
Status
Deployme...
Deployme...
Revision lo...
Initiating e...
Star
No results
There are no results to display.

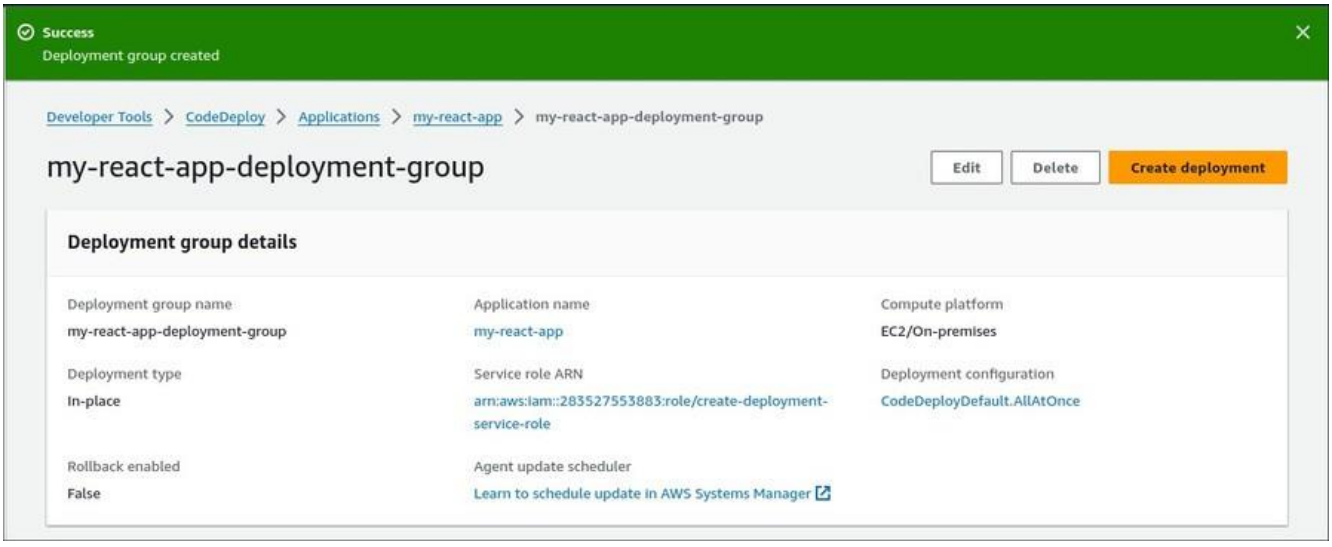

Policy name
Type
Attached entities
AmazonEC2FullAccess
AWS managed
3
AmazonEC2RoleforAWSCodeDeploy
AWS managed
1
AmazonEC2RoleforAWSCodeDeployLi...
AWS managed
1
AmazonS3FullAccess
AWS managed
3
AWSCodeDeployFullAccess
AWS managed
1
AWSCodeDeployRole
AWS managed
1
CodeBuildBasePolicy-react-app-build-rol...
Customer managed
1

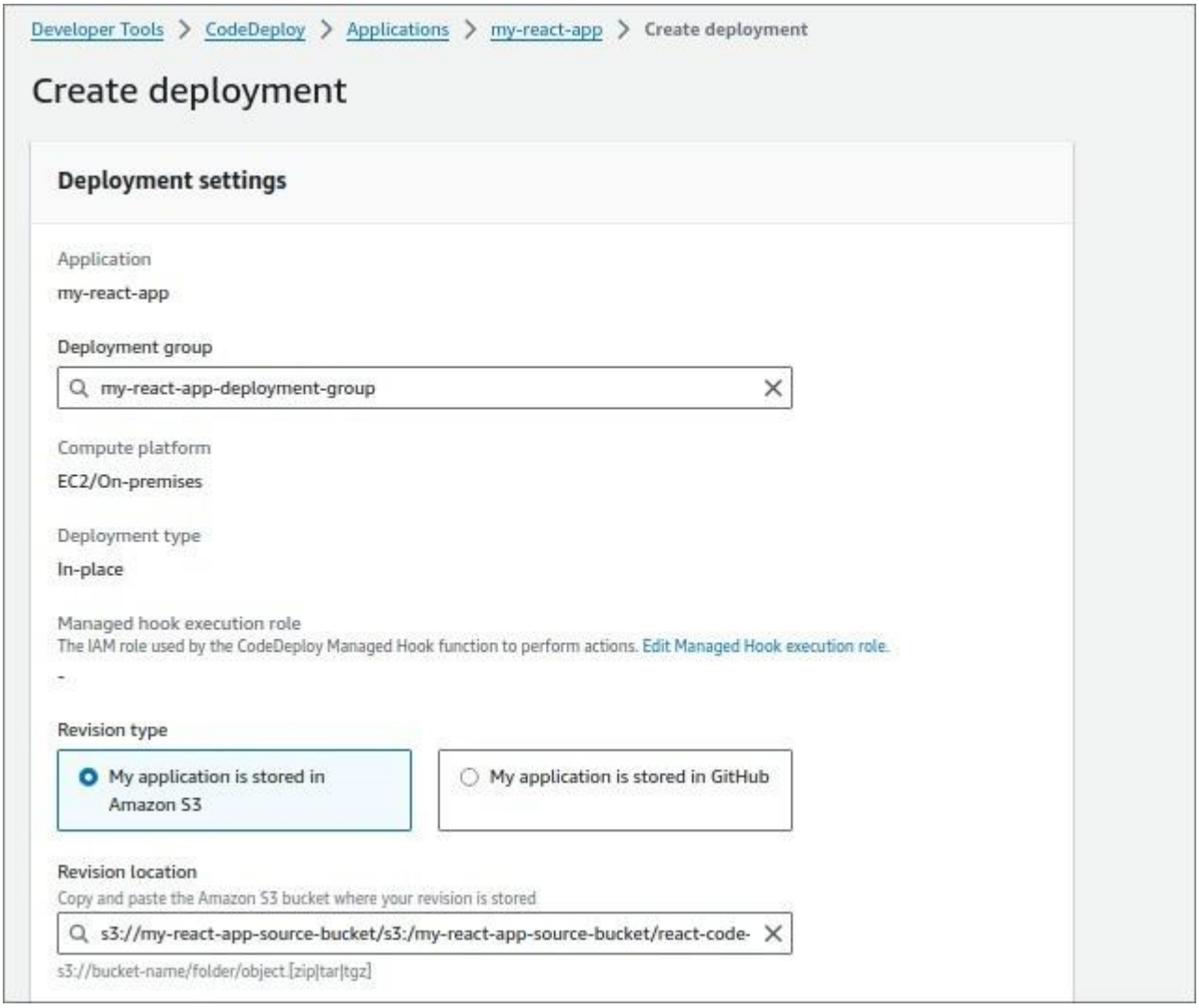

EC2 > Instances > Launch an instance
Launch an instance Info
Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.
Name and tags Info
Name
my-react-app-ec2
Add additional tags
Application and OS Images (Amazon Machine Image) Info
An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below
Search our full catalog including 1000s of application and OS images
Recents
Quick Start
Summary
Number of instances Info
1
Software Image (AMI)
Amazon Linux 2023 AMI 2023.5.2...read more
ami-0ba9883b710b05ac6
Virtual server type (instance type)
t2.micro
Firewall (security group)
all
Storage (volumes)
1 volume(s) - 8 GiB
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in
Cancel
Launch instance
Review commands

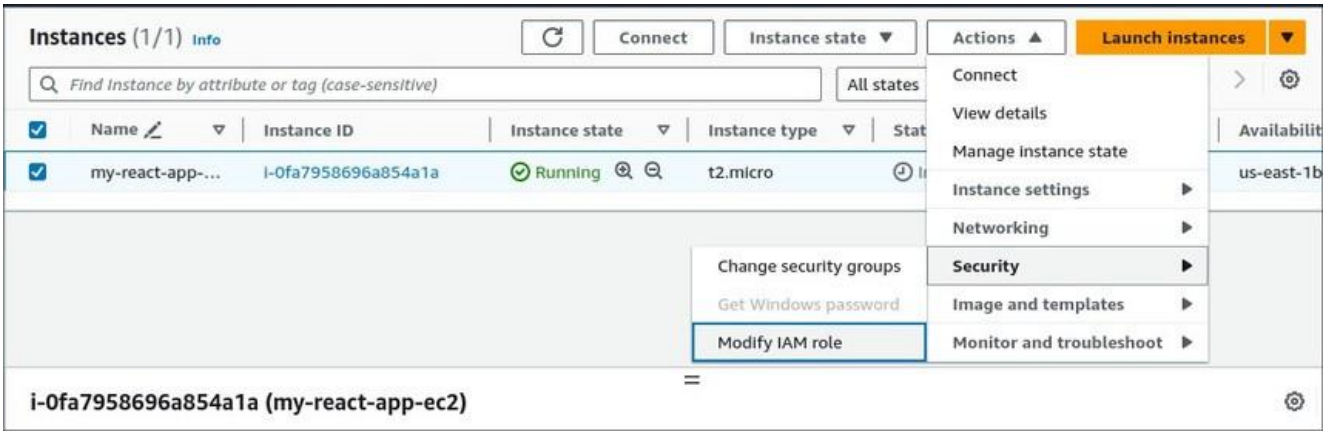

Developer Tools > CodeDeploy > Applications > my-react-app > Create deployment group
Create deployment group
Application
Application
my-react-app
Compute type
EC2/On-premises
Deployment group name
Enter a deployment group name
my-react-app-deployment-group
100 character limit
Service role
Enter a service role
Enter a service role with CodeDeploy permissions that grants AWS CodeDeploy access to your target instances.
arn:aws:iam::283527553883:role/service-role/react-app-build-roleee

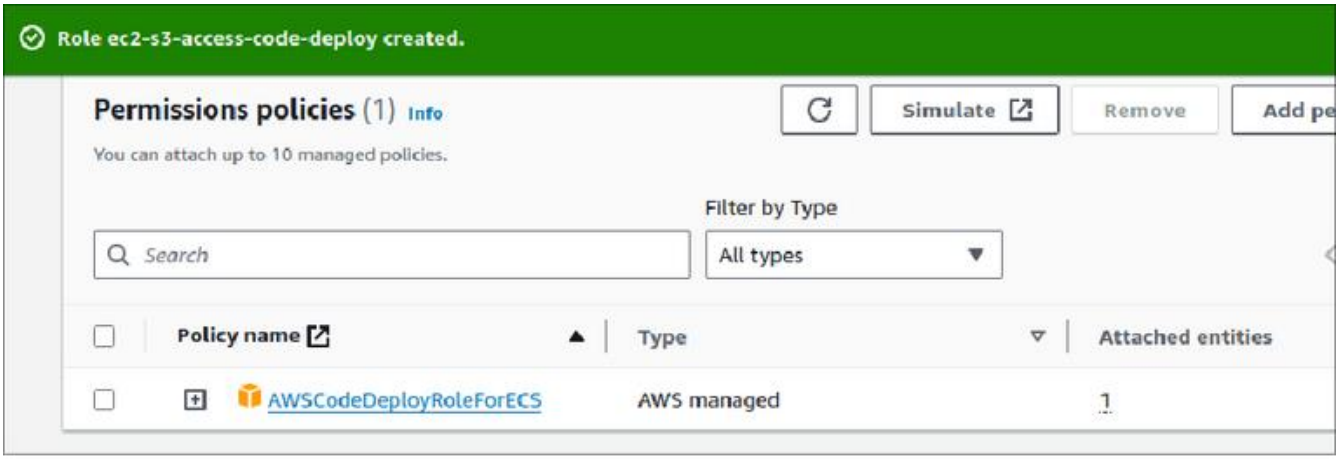

Environment configuration
Select any combination of Amazon EC2 Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add to this deployment
Amazon EC2 Auto Scaling groups
Amazon EC2 instances
1 unique matched instance. Click here for details
You can add up to three groups of tags for EC2 instances to this deployment group.
One tag group: Any instance identified by the tag group will be deployed to.
Multiple tag groups: Only instances identified by all the tag groups will be deployed to.
Tag group 1
Key
Value - optional
Name
my-react-app-ec2
Remove tag
Add tag
Add tag group
On-premises instances
Matching instances
1 unique matched instance. Click here for details

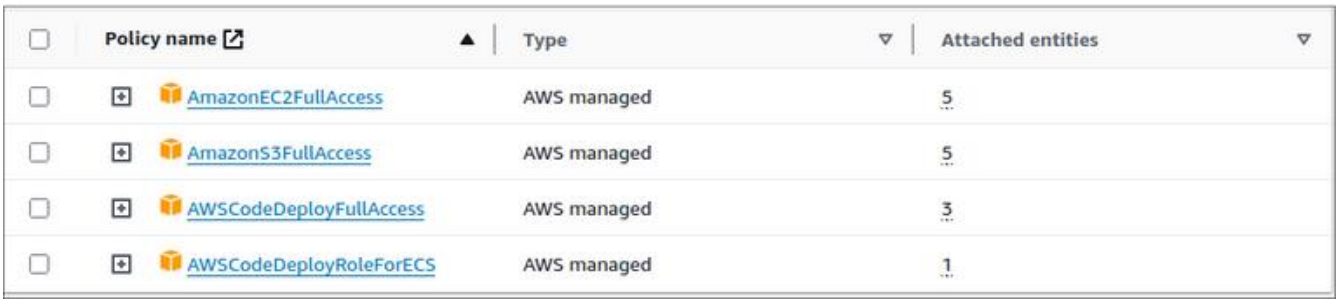
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ec2.amazonaws.com",
                    "codedeploy.amazonaws.com"
                    ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

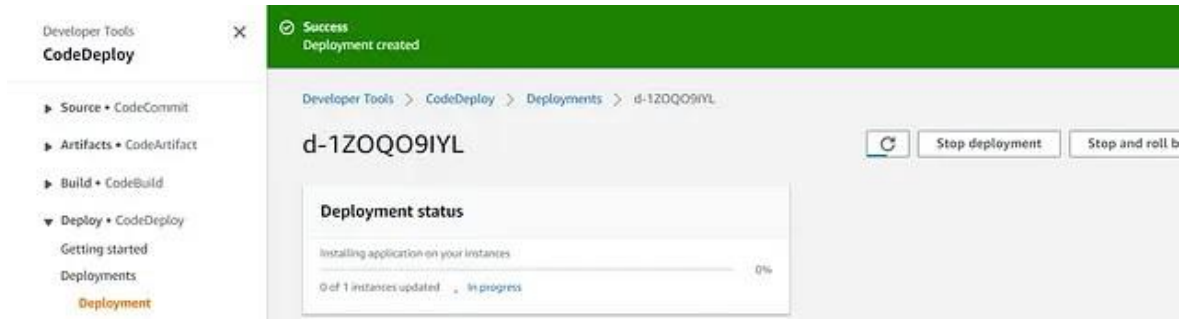


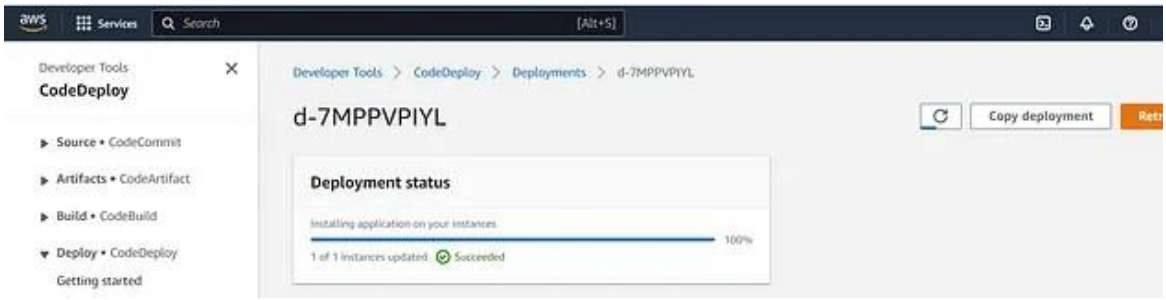

Developer Tools > CodeDeploy > Applications > my-react-app > Create deployment
Create deployment
Deployment settings
Application
my-react-app
Deployment group
my-react-app-deployment-group
Compute platform
EC2/On-premises
Deployment type
In-place
Managed hook execution role
The IAM role used by the CodeDeploy Managed Hook function to perform actions. Edit Managed Hook execution role.
-
Revision type
My application is stored in Amazon S3
My application is stored in GitHub
Revision location
Copy and paste the Amazon S3 bucket where your revision is stored
s3://my-react-app-source-bucket/s3:/my-react-app-source-bucket/react-code-
s3://bucket-name/folder/object.[zip|tar|tgz]

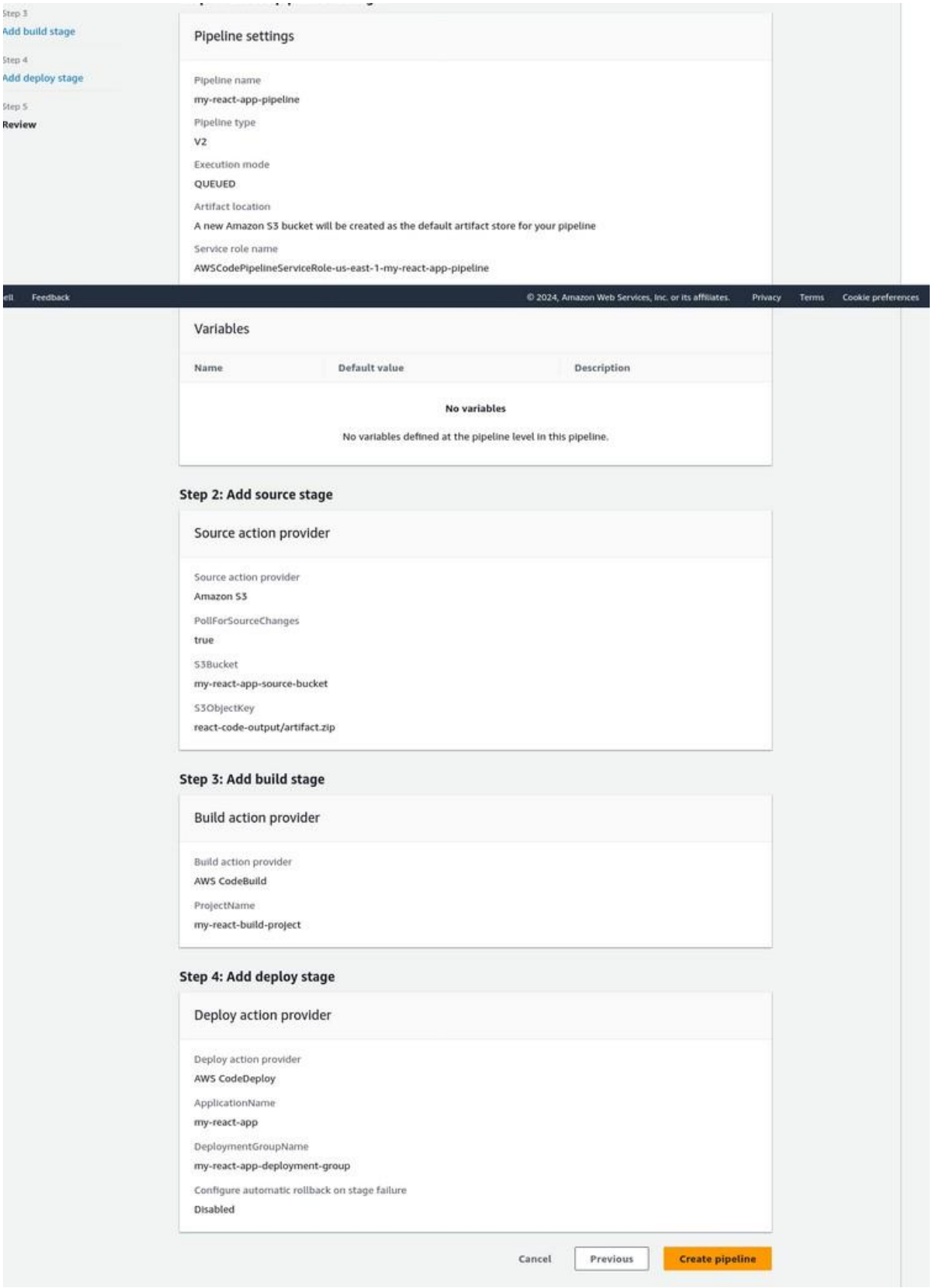

Instances (1/1) Info
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Availabilit
my-react-app-...
i-0fa7958696a854a1a
Running
t2.micro
us-east-1b
Connect
View details
Manage instance state
Instance settings
Networking
Security
Image and templates
Monitor and troubleshoot
Change security groups
Get Windows password
Modify IAM role
i-0fa7958696a854a1a (my-react-app-ec2)

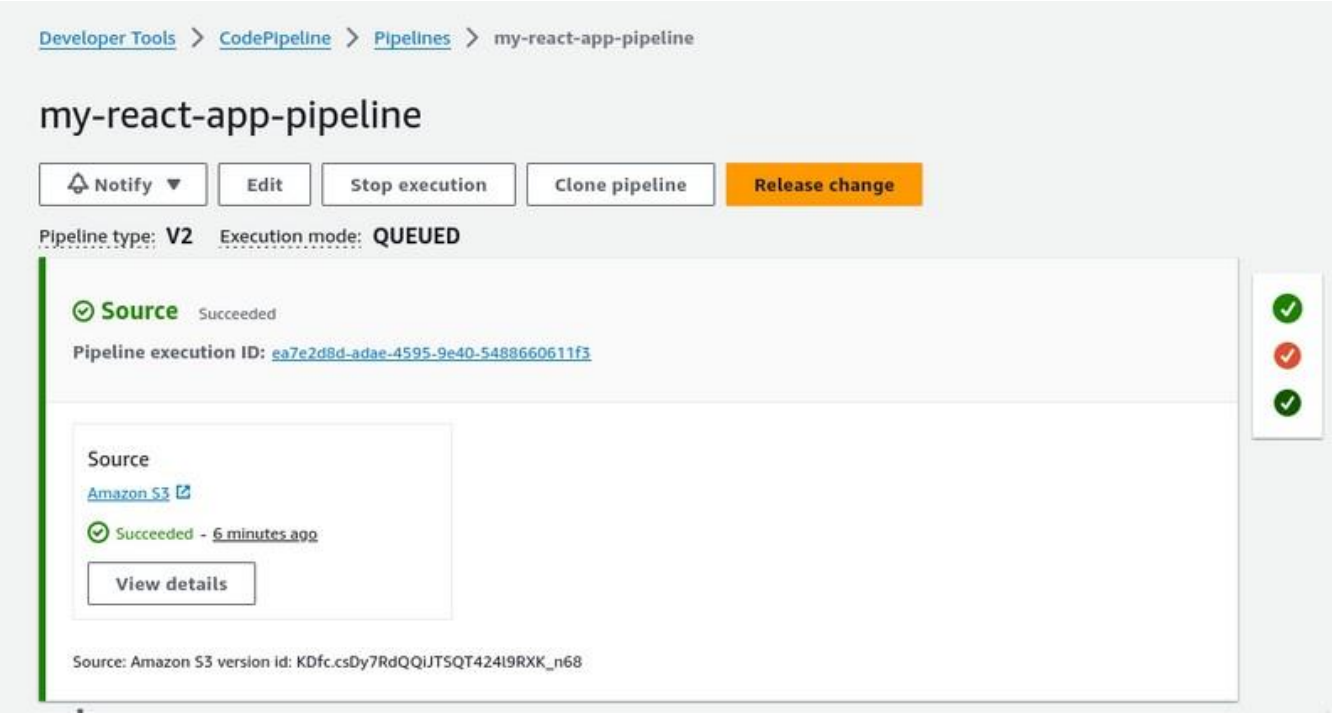

Role ec2-s3-access-code-deploy created.
Permissions policies (1) Info
You can attach up to 10 managed policies.
Simulate
Remove
Add pe
Filter by Type
Search
All types
Policy name
Type
Attached entities
AWSCodeDeployRoleForECS
AWS managed
1


Policy name
Type
Attached entities
AmazonEC2FullAccess
AWS managed
5
AmazonS3FullAccess
AWS managed
5
AWSCodeDeployFullAccess
AWS managed
3
AWSCodeDeployRoleForECS
AWS managed
1


Developer Tools
CodeDeploy
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Getting started
Deployments
Deployment
Success
Deployment created
Developer Tools > CodeDeploy > Deployments > d-1ZOQO9IYL
d-1ZOQO9IYL
Stop deployment
Deployment status
0%
0 of 1 instances updated
In progress


Services
Search
[Alt+S]
Developer Tools
CodeDeploy
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Getting started
Developer Tools > CodeDeploy > Deployments > d-7MPPVPIYL
d-7MPPVPIYL
Copy deployment
Deployment status
100%
1 of 1 instances updated
Succeeded


Step 3
Add build stage
Step 4
Add deploy stage
Step 5
Review
Pipeline settings
Pipeline name
my-react-app-pipeline
Pipeline type
V2
Execution mode
QUEUED
Artifact location
A new Amazon S3 bucket will be created as the default artifact store for your pipeline
Service role name
AWSCodePipelineServiceRole-us-east-1-my-react-app-pipeline
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Variables
Name
Default value
Description
No variables
No variables defined at the pipeline level in this pipeline.
Step 2: Add source stage
Source action provider
Source action provider
Amazon S3
PollForSourceChanges
true
S3Bucket
my-react-app-source-bucket
S3ObjectKey
react-code-output/artifact.zip
Step 3: Add build stage
Build action provider
Build action provider
AWS CodeBuild
ProjectName
my-react-build-project
Step 4: Add deploy stage
Deploy action provider
Deploy action provider
AWS CodeDeploy
ApplicationName
my-react-app
DeploymentGroupName
my-react-app-deployment-group
Configure automatic rollback on stage failure
Disabled
Cancel
Previous
Create pipeline



Conclusion: Thus, we have successfully Build our Application using AWS Code Build and Deploy on S3 / SEBS using AWS Code Pipeline, deployed Sample Application on EC2 instance using AWS Code Deploy.

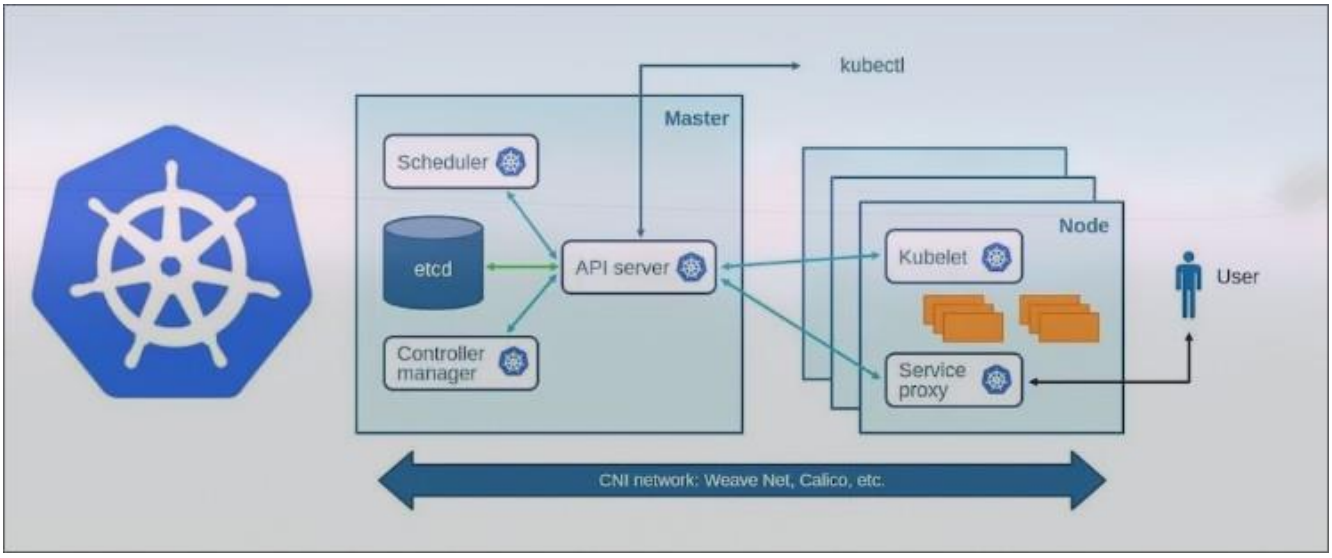
EXPERIMENT - 3

Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

Theory:

Kubernetes

Kubernetes is an open source platform for managing container technologies such as Docker. Docker lets you create containers for a pre-configured image and application. Kubernetes provides the next step, allowing you to balance loads between containers and run multiple containers across multiple systems.



Kubernetes Architecture:

Kubernetes is consists of two nodes master & worker, Nodes are the physical or virtual machines that are used to run pods. There can be only one master and multiple worker nodes.

- Master (Control plane)

Master node often reffered to as control plane and responsible for managing and orchestrating the overall operations of the system, It serves as central control point of cluster. It interact with worker node to deploy pods.

Components:

- Schedhuler: schedule worker node for running pods.
- Controller manager: The main function is to maintain desired state of cluster. Checks what the workers are doing and they are up.
- API server: Directly communicated with worker from master and vice versa.
- etcd: Store state of kubernetes cluster in key-value data store.
- Kubectl: Kubectl is a command-line tool used to communicate with a Kubernetes cluster's control plane using the API server. Allows to run commands to deploy application, inspect and manage resources.
- CNI: Container network interface.
- Pod: This host and manage our containers that run our application.

- Worker:

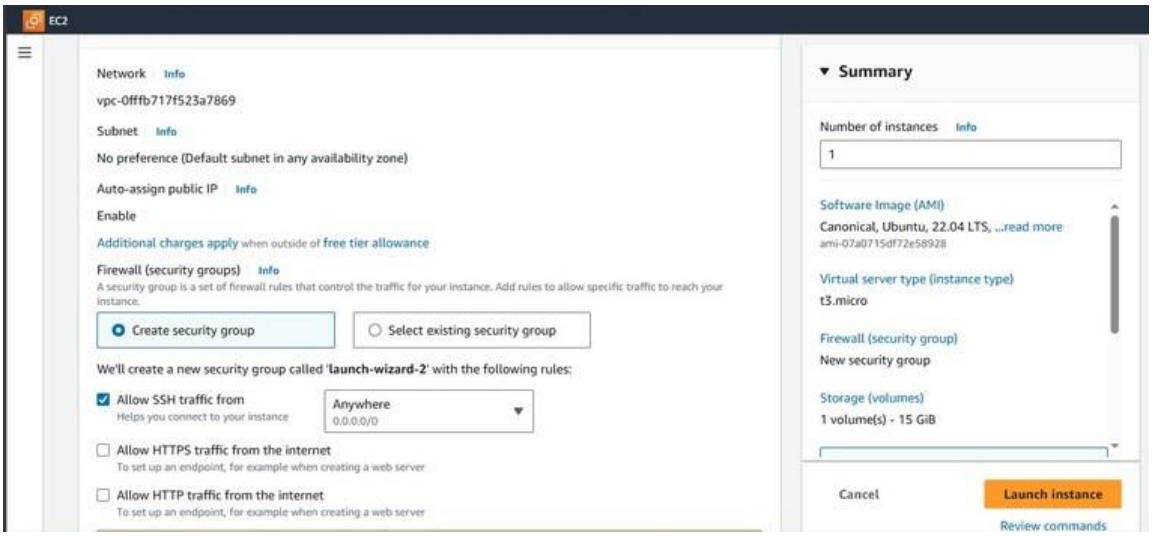
Machine which runs containers and workloads. This is where the actual application is running.

Components:

- kubelet: Stay in worker node, Manage containers and ensures they're running as expected. It communicates with the API server to receive information about the pods that are assigned to the node.
- service proxy : It maintains some network rules which determines how traffic is allowed to and from the Pods. It also Allow users/clients to access application.

Steps:

1. Start by creating two Aws EC2 instance for worker & master node



2. After creating servers execute commands according on master and worker:

- Install Dependencies *# Execute On both the nodes (master and worker)*

~$ sudo apt update

~$ sudo apt-get install -y apt-transport-https ca-certificates curl

~$ sudo apt install docker.io -y

~$ sudo systemctl enable --now docker

- Install kubeadm # Execute On both the nodes (master and worker)

~$ echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https:-/pkgs.k8s.io/core:/stable:/v1.28/deb/ /" | sudo tee /etc/apt/sources.list.d/kubernetes.list

~$ curl -fsSL https:-/pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
~$ sudo apt update

~$ sudo apt install kubeadm kubectl kubelet -y

- Now, initialize kubeadm (kubernetes) in Master node.

~$ sudo kudeadm init

- Setup local kubeconfig # Execute on master node.

~$ mkdir -p $HOME/.kube

~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config

- Apply weave network # On both

~$ kubectl apply -f https:-/github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml

- Generate token for worker node to join # On master

~$ sudo kubeadm token create –print-join-command

- Now, paste the token (output) of 6th step, also append --v=5 at end.

~$ sudo kubeadm join 172.31.61.228:6443 --token f4mesu.7rzk86ga48n3uydh --discovery-token-ca-cert-hash sha256:66c9863913ffdb50316e82b74f3703f73e42c4210d3a01ec7afcdbc01f677eec –v=5

```
ubuntu@ip-172-31-31-164: ~
ubuntu@ip-172-31-31-164:~$ kubectl apply -f https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.apps/weave-net created
ubuntu@ip-172-31-31-164:~$ sudo kubeadm token create --print-join-command
kubeadm join 172.31.31.164:6443 --token rg7l3c.rcc3916mnkrguhm4 --discovery-token-ca-cert-hash sha256:f58383dec71f592f3095416c536bf7d98b0b4796a3d193d36d0534a879219d19
ubuntu@ip-172-31-31-164:~$ sudo kubeadm token create --print-join-command

timed out waiting for the condition
To see the stack trace of this error execute with --v=5 or higher
ubuntu@ip-172-31-31-164:~$ history
    1  docker ps
    2  echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /" | sudo tee /etc/apt/sources.list.d/kubernetes
.list
    3  curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
    4  sudo apt-get update
    5  sudo apt install kubeadm=1.20.0-00 kubectl=1.20.0-00 kubelet=1.20.0-00 -y
    6  sudo apt install kubeadm kubectl kubelet -y
    7  sudo kudeadm init
    8  kubeadm init
    9  sudo kubeadm init
   10  mkdir -p $HOME/.kube
   11  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
   12  sudo chown $(id -u):$(id -g) $HOME/.kube/config
   13  kubectl apply -f https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml
   14  export KUBECONFIG=/etc/kubernetes/admin.conf
   15  kubectl apply -f https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml
   16  id -u
   17  sudo chown $USER: ~/.kube/config
   18  kubectl apply -f https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml
   19  docker ps
   20  cat /etc/passwd | less
   21  kubectl apply -f https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml
   22  ls -la ~/.kube/config
   23  sudo chown $(id -u):$(id -g) $HOME/.kube/config
   24  kubectl apply -f https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml
   25  export KUBECONFIG=
   26  kubectl apply -f https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml
   27  sudo kubeadm token create --print-join-command
   28  history
ubuntu@ip-172-31-31-164:~$
```

- Verify you worker node connection by running this command on Master node.

~$ kubectl get pods

```
ubuntu@ip-172-31-29-127:~$ sudo kubeadm join 172.31.31.164:6443 --token 7tivfy.u60211j435ymnukt --discovery-token-ca-cert-hash sha256:a7ca0f7539bab4f1f2f32e
5e4da6c5309b231540276061989d8a04409b22359e
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

ubuntu@ip-172-31-29-127:~$
```

Conclusion: Thus we successfully understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

EXPERIMENT - 4

Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

Theory:

After successfully creating the Kubernetes cluster, you can now deploy your application to it using the kubectl.

You need to create manifest's for your project.

- Deployment.yaml : You actual app deploying

- service.yaml : To access your app outside the network in public.

Before moving forward you can install kubectl in your system using package manager:

□ ~ □ sudo pacman -S kubectl

Some common terms in manifest:

In Kubernetes manifests, you'll encounter several common terms and concepts. Here's a brief overview of some key ones:

- kind: Should be Deployment.
- apiVersion: Specifies the API version, e.g., apps/v1.
- metadata: Contains the name, namespace, labels, and annotations of the deployment.
- spec: The specification for the deployment.
- replicas: Number of pod replicas to run.
- selector: Defines how to select pods managed by the deployment.
- matchLabels: Key-value pairs for selecting pods.
- template: The pod template that describes the pods.
  - metadata: Labels for the pods.
  - spec: Defines the containers and their configurations.
    - containers: List of container specs.
      - name: Name of the container.

- image: Container image to use.
- ports: List of ports to expose.
- env: Environment variables for the container.
- volumeMounts: Specifies where to mount volumes in the container.
-

Service manifest:

- apiVersion: Typically v1.
- kind: Should be Service.
- metadata: Contains the name, namespace, labels, and annotations of the service.
- spec: The specification for the service.
-

- selector: Defines which pods the service will target based on labels.
- ports: List of ports exposed by the service.
    - port: Port that the service will expose.
    - targetPort: Port on the container to forward traffic to.
- type: Defines the service type (e.g., ClusterIP, NodePort, LoadBalancer).

Here's mine:

deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name:  node-deployment
 namespace: thrifty
 labels:
      app: node-api
spec:
 replicas: 2
 selector:
      matchLabels:
         app: node-api
 template:
       metadata:
            labels:
             app: node-api
      spec:
         containers:
         - name: node-api
            image: darkkernel/node-api
            ports:
            - containerPort:  8080
```

Service.yaml

```
apiVersion: v1
kind: Service
metadata:
 name:  external-svc
 namespace: thrifty
 labels:
      app: external-svc
spec:
 type: LoadBalancer
 ports:
      - port: 80
        targetPort: 8080
        protocol: TCP
 selector:
      app:  node-api
```

1. Now let's start with deployment, check if the cluster is ready.

```
□ ~ □ kubectl get nodes
NAME                                STATUS   ROLES    AGE     VERSION   v1.30.2-eks-
ip-192-168-26-47.ec2.internal       Ready    <none>   5m32s   1552ad0   v1.30.2-eks-
ip-192-168-33-0.ec2.internal        Ready    <none>   5m43s   1552ad0
```

2. Get your project in your system.

```
> git clone https://github.com/Dark-Kernel/node-api.git
Cloning into 'node-api'//.
remote: Enumerating objects: 1513, done.
remote: Counting objects: 100% (337/337), done.
remote: Compressing objects: 100% (194/194), done.
remote: Total 1513 (delta 126), reused 337 (delta 126), pack-reused 1176 (from 1)
Receiving objects: 100% (1513/1513), 2.16 MiB | 5.71 MiB/s, done.
Resolving deltas: 100% (357/357), done.
>
> cd  node-api/
> ls Kubernetes/
□ deployment.yaml □ service.yaml
>
```

3. Create Namespace if required

```
> kubectl create namespace thrifty
namespace/thrifty created
```

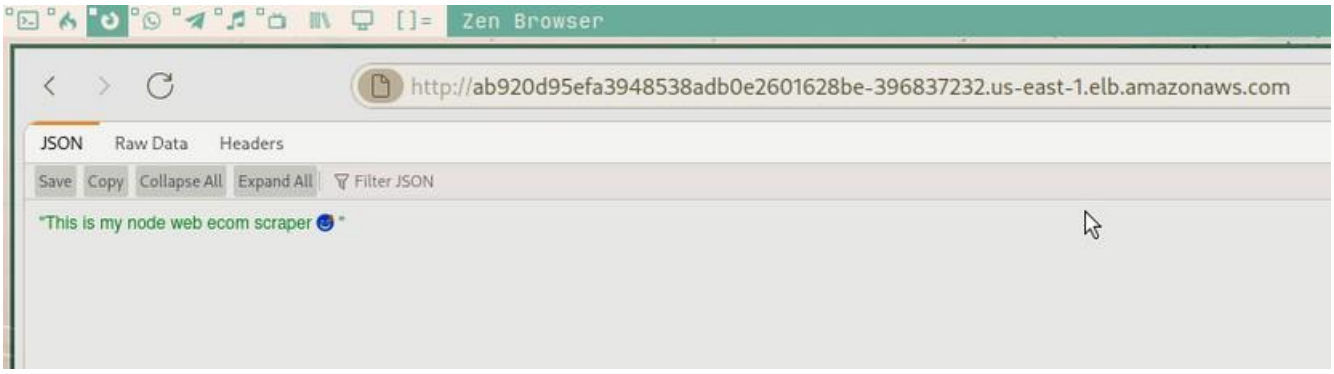
4. Now, create the deployment using the kubectl command.
> kubectl apply -f Kubernetes/deployment.yaml
deployment.apps/node-deployment created
5. You can check if it is applied.
> kubectl get deployments -n thrifty

NAME	READY	UP-TO-DATE	AVAILABLE	AG
node-deployment	2/2	2	2	E
>				27s

6. Then deploy the service
> kubectl apply -f Kubernetes/external-svc.yaml
service/external-svc created
7. Now, you can check your service if it is done.
> kubectl get svc -n thrifty

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	
PORT(S)	AGE			
external-svc	LoadBalancer	10.100.232.57	ab920d95efa3948538adb0e2601628be-396837232.us-	
east-1.elb.amazonaws.com		80:31713/TCP	107s	

Now using the external IP, which is a subdomain actually you can access your application.
And our application deployment is successful



Conclusion: Thus, we have successfully installed Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deployed our First Kubernetes Application.

Experiment no. 5

Aim : To understand terraform cycle, core concepts and terminologies and install it on Linux machine.

Theory :

- **Terraform lifecycle**

The lifecycle of a resource managed by Terraform has three stages: Apply, Update, and Destroy. The lifecycle meta-argument allows users to control these stages.

- **Terraform CLI**

The Terraform Command Line Interface (CLI) is the primary tool for interacting with Terraform's code.

- **Terraform core workflow**

The core workflow of Terraform consists of the following stages:

1. init: Initializes the local Terraform environment
2. plan: Compares the Terraform state with the cloud state, and creates an execution plan
3. apply: Executes the plan to change the deployment
4. destroy: Removes all resources defined in the Terraform configuration

- **Core concepts**

Some other core concepts of Terraform include:

Variables: Key-value pairs that allow customization

Provider: A plugin that allows users to interact with APIs of a service

Module: A folder containing Terraform templates that define configurations

State: Cached information about the infrastructure managed by Terraform

Resources: A block of one or more infrastructure objects

Data Source: Returns information on external objects to Terraform

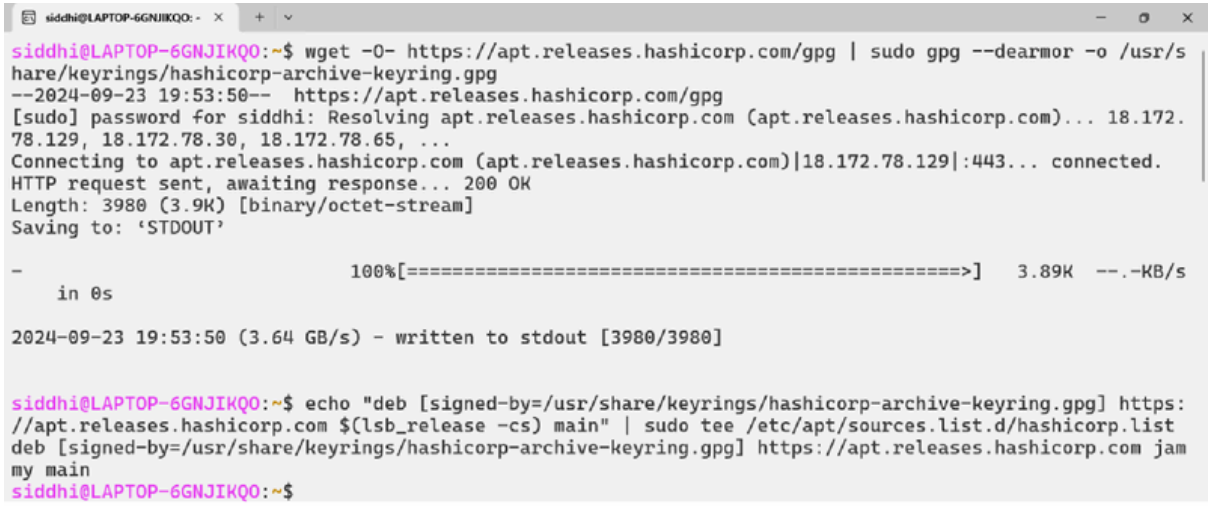
Output Values: Return values of a Terraform module that can be used by other configurations

Output :

```
wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg


echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list


sudo apt update && sudo apt install terraform
```



```
siddhi@LAPTOP-6GNJIKQO:~$ wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg
--2024-09-23 19:53:50--  https://apt.releases.hashicorp.com/gpg
[sudo] password for siddhi: Resolving apt.releases.hashicorp.com (apt.releases.hashicorp.com)... 18.172.78.129, 18.172.78.30, 18.172.78.65, ...
Connecting to apt.releases.hashicorp.com (apt.releases.hashicorp.com)|18.172.78.129|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3980 (3.9K) [binary/octet-stream]
Saving to: 'STDOUT'

-                           100%[=================================================>]   3.89K  --.-KB/s    in 0s

2024-09-23 19:53:50 (3.64 GB/s) - written to stdout [3980/3980]


siddhi@LAPTOP-6GNJIKQO:~$ echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list
deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com jammy main
siddhi@LAPTOP-6GNJIKQO:~$
```

```
siddhi@LAPTOP-6GNJIKQO:~$ sudo apt update && sudo apt install terraform
Get:1 https://apt.releases.hashicorp.com jammy InRelease [12.9 kB]
Get:2 https://apt.releases.hashicorp.com jammy/main amd64 Packages [150 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Hit:4 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:5 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1844 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [298 kB]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [13.3 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [2439 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2062 kB]
Get:12 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [420 kB]
Get:13 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 c-n-f Metadata [584 B]
Get:14 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [905 kB]
Get:15 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [177 kB]
Get:16 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [19.3 kB]
Get:17 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [37.2 kB]
Get:18 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 c-n-f Metadata [228 B]
Get:19 http://archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [356 kB]
Get:20 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [17.8 kB]
Get:21 http://archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [2499 kB]
Get:22 http://archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [430 kB]
Get:23 http://archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 c-n-f Metadata [616 B]
Get:24 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1125 kB]
Get:25 http://archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [262 kB]
Get:26 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metadata [26.1 kB]
Get:27 http://archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [43.3 kB]
Get:28 http://archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 c-n-f Metadata [444 B]
Get:29 http://archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [67.8 kB]
Get:30 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [28.8 kB]
```

```
Get:31 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 c-n-f Metadata [672 B]
Fetched 13.6 MB in 4s (3261 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
134 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  terraform
0 upgraded, 1 newly installed, 0 to remove and 134 not upgraded.
Need to get 28.1 MB of archives.
After this operation, 89.1 MB of additional disk space will be used.
Get:1 https://apt.releases.hashicorp.com jammy/main amd64 terraform amd64 1.9.6-1 [28.1 MB]
Fetched 28.1 MB in 6s (4892 kB/s)
Selecting previously unselected package terraform.
(Reading database ... 24606 files and directories currently installed.)
Preparing to unpack .../terraform_1.9.6-1_amd64.deb ...
Unpacking terraform (1.9.6-1) ...
Setting up terraform (1.9.6-1) ...
siddhi@LAPTOP-6GNJIKQO:~$
```

```
siddhi@LAPTOP-6GNJIKQO:~$ terraform
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init          Prepare your working directory for other commands
  validate      Check whether the configuration is valid
  plan          Show changes required by the current configuration
  apply         Create or update infrastructure
  destroy       Destroy previously-created infrastructure

All other commands:
  console       Try Terraform expressions at an interactive command prompt
  fmt           Reformat your configuration in the standard style
  force-unlock  Release a stuck lock on the current workspace
  get           Install or upgrade remote Terraform modules
  graph         Generate a Graphviz graph of the steps in an operation
  import        Associate existing infrastructure with a Terraform resource
  login         Obtain and save credentials for a remote host
  logout        Remove locally-stored credentials for a remote host
  metadata      Metadata related commands
  output        Show output values from your root module
  providers     Show the providers required for this configuration
  refresh       Update the state to match remote systems
  show          Show the current state or a saved plan
  state         Advanced state management
  taint         Mark a resource instance as not fully functional
  test          Execute integration tests for Terraform modules
  untaint       Remove the 'tainted' state from a resource instance
```

VASANTDADA PATIL PRATISHTHAN'S COLLEGE OF ENGINEERING AND VISUAL ARTS
ISO 9001:2015 Certified Institute
Department of Information Technology
NBA Accredited Course (Dated 01/07/2024 to 30/06/2027)

EXPERIMENT - 6

Aim: To Build, change, and destroy AWS / GCP /Microsoft Azure/ Digital Ocean infrastructure Using Terraform.

Theory:

Infrastructure as code (IaC) tools allow you to manage infrastructure with configuration files rather than through a graphical user interface. IaC allows you to build, change, and manage your infrastructure in a safe, consistent, and repeatable way by defining resource configurations that you can version, reuse, and share.

Terraform is HashiCorp's infrastructure as code tool. It lets you define resources and infrastructure in human-readable, declarative configuration files, and manages your infrastructure's lifecycle. Using Terraform has several advantages over manually managing your infrastructure.

Terraform can manage infrastructure on multiple cloud platforms. The human-readable configuration language helps you write infrastructure code quickly. Terraform's state allows you to track resource changes throughout your deployments. You can commit your configurations to version control to safely collaborate on infrastructure.
Manage any infrastructure Terraform plugins called providers let Terraform interact with cloud platforms and other services via their application programming interfaces (APIs). HashiCorp and the Terraform community have written over 1,000 providers to manage resources on Amazon Web Services (AWS), Azure, Google Cloud Platform (GCP), Kubernetes, Helm, GitHub, Splunk, and DataDog, just to name a few. Find providers for many of the platforms and services you already use in the Terraform Registry. If you don't find the provider you're looking for, you can write your own.

To deploy infrastructure with Terraform:

- Scope - Identify the infrastructure for your project.
- Author - Write the configuration for your infrastructure.
- Initialize - Install the plugins Terraform needs to manage the infrastructure.
- Plan - Preview the changes Terraform will make to match your configuration.
- Apply - Make the planned changes. Build Infrastructure

Prerequisites
Before we proceed we need some thing to be setup.

- The Terraform installed.
- The AWS CLI installed.
- An AWS account.
- AWS access key credentials.

1. We can configure aws cli

> aws configure
AWS Access Key ID [None]: AKIAUEA4PENWWEXX52MW
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]:

2. Then go with terraform, create a directory in which main.tf will live

> mkdir myTerraInfra

3. Create main.tf which contains our main configuration for infrastructure

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = " > 4.16"
    }
  }
  required_version = " = 1.2.0"
}

provider "aws" { region =
  "us-east-1"
}

resource "aws_instance" "app_server" { ami   =
  "ami-0453898e98046c639"
  instance_type = "t2.micro"

  tags = {
    Name = var.instance_name
  }

  provisioner "remote-exec" { inline = [
      "sudo apt-get -y update",
      "sudo apt-get -y install nginx", "sudo
      service nginx start"
    ]
  }
}
```

4. Once the configuration file is created, initialize the terraform so It can install required providers defined in main.tf

> terraform init Initializing the
backend .
Initializing provider plugins .
- Finding hashicorp/aws versions matching " > 4.16" .
- Installing hashicorp/aws v4.67.0 .
- Installed hashicorp/aws v4.67.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider selections it made above. Include this file in your version control repository so that Terraform can guarantee to make the same selections by default when you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see any changes that are required for your infrastructure.

5. Now, we can go with formatting and validating our configuration files.

> terraform fmt
> terraform validate
Success! The configuration is valid

6. Next is the main part, here we create actual infrastructure

> terraform apply

It will show dry run plan and will ask to continue

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

After this it will create your infrastructure, you can verify by login to aws console. Or using aws cli.

You can modify you code and rerun this to update your infrastructure. There can be different files like variables.tf, or anyother.

When the infrastructure building gets completed, 2 files will be generated:

- terraform.tfstate : It stores the status of current infrastructure applied
- terrafrom.tfstate.backup : It is backup if above file incase of mess.
- .terraform : This directory contains the providers packages.

7. Now to destroy it run the following command

> terraform destroy

It will again ask to continue, type yes and It will start destroying

.
aws_instance.app_server: Destroying . [id=i-04701c6a4dd4c7364] aws_instance.app_server: Still destroying . [id=i-04701c6a4dd4c7364, 10s elapsed] aws_instance.app_server: Still destroying . [id=i-04701c6a4dd4c7364, 20s elapsed] aws_instance.app_server: Still destroying . [id=i-04701c6a4dd4c7364, 30s elapsed] aws_instance.app_server: Still destroying . [id=i-04701c6a4dd4c7364, 40s elapsed] aws_instance.app_server: Still destroying . [id=i-04701c6a4dd4c7364, 50s elapsed] aws_instance.app_server: Still destroying . [id=i-04701c6a4dd4c7364, 1m0s elapsed] aws_instance.app_server: Destruction complete after 1m2s
Destroy complete! Resources: 1 destroyed.

Conclusion: Thus, we have successfully Build, changed, and destroyed AWS / GCP /Microsoft Azure/ Digital Ocean infrastructure Using Terraform.

EXPERIMENT - 7

Aim: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to Sonar Qube/Git Lab.

Theory:

- SAST:

Static Application Security Testing is an automated process that analyzes source code or compiled versions of code to identify potential security vulnerabilities. It's "static" because it examines the code without executing it.

- Jenkins:

Jenkins is an open-source automation server that enables developers to reliably build, test, and deploy their software. It is a server-based system that runs in servlet containers such as Apache Tomcat. It is an CI/CD tool used in most of the industries.

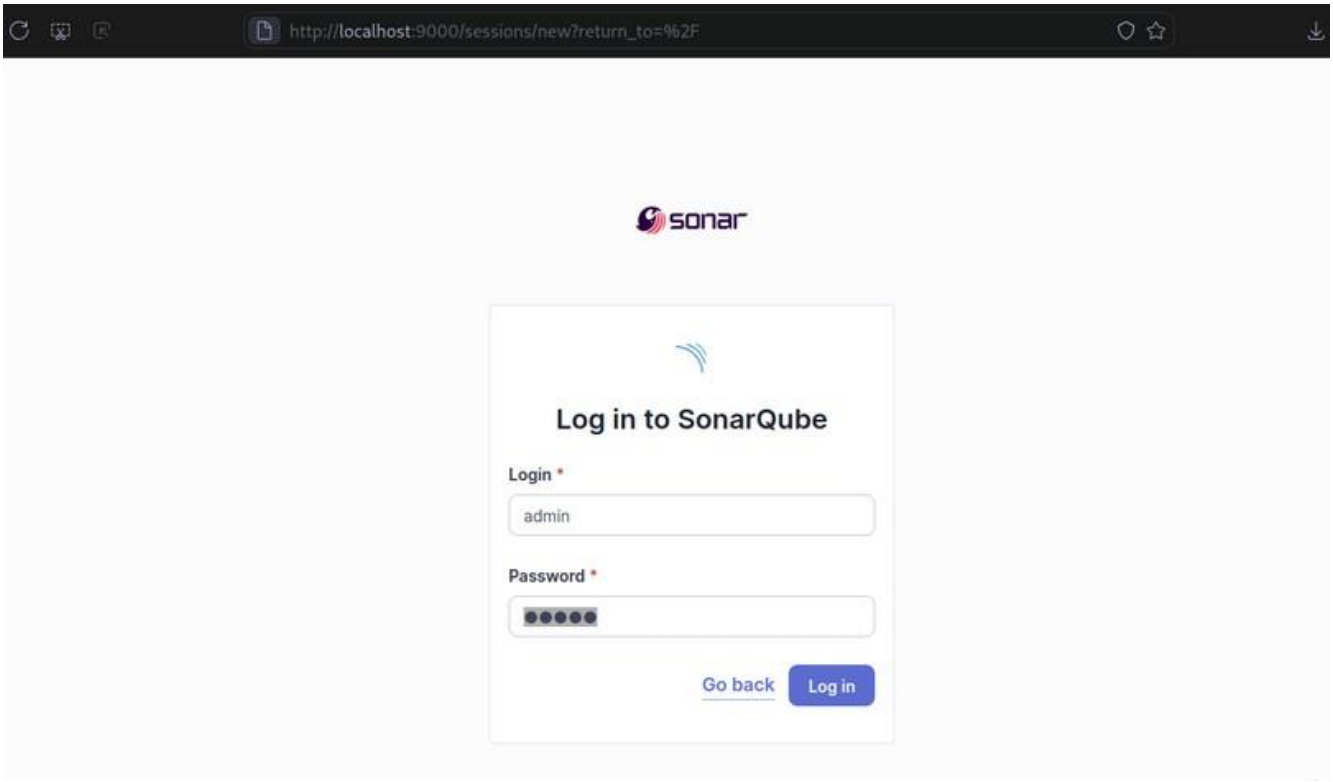
- SonarQube:

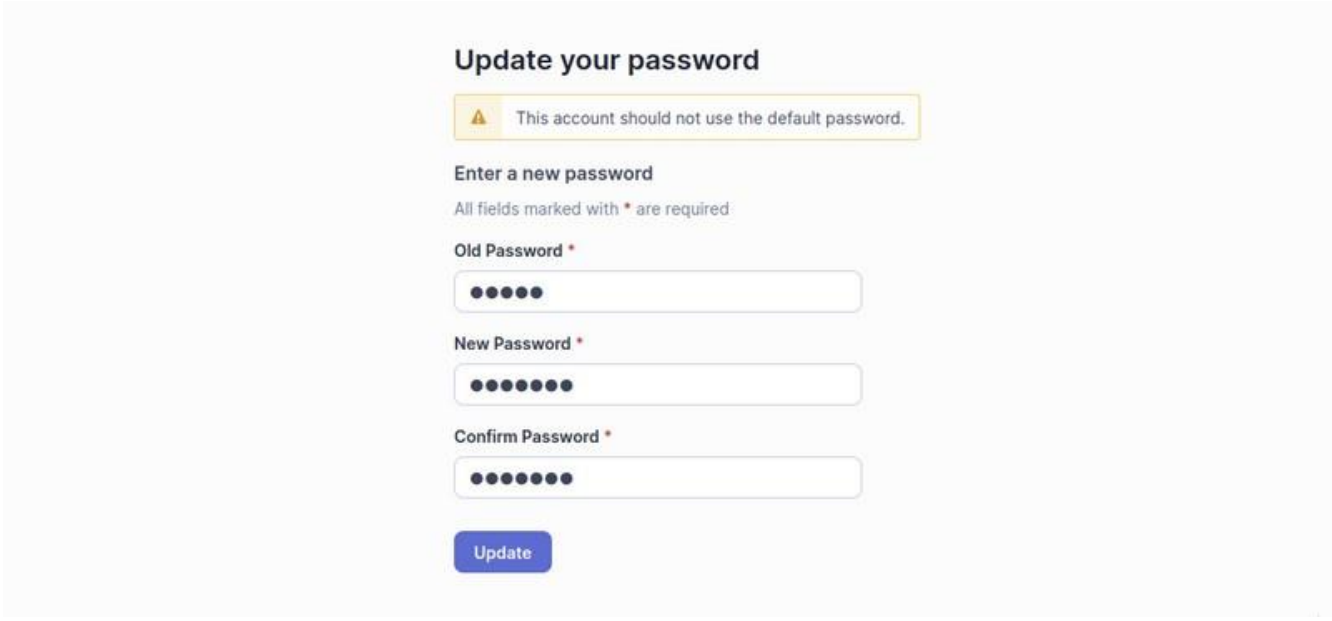
SonarQube is an open-source platform for continuous inspection of code quality, developed by SonarSource. It provides automated analysis and integration with various development tools, enabling developers to write clean, readable, and understandable code.

Integrating SAST into Jenkins allows for automated security checks as part of your build process. Also It can be integrated with SonarQube to provide detailed code analysis reports.

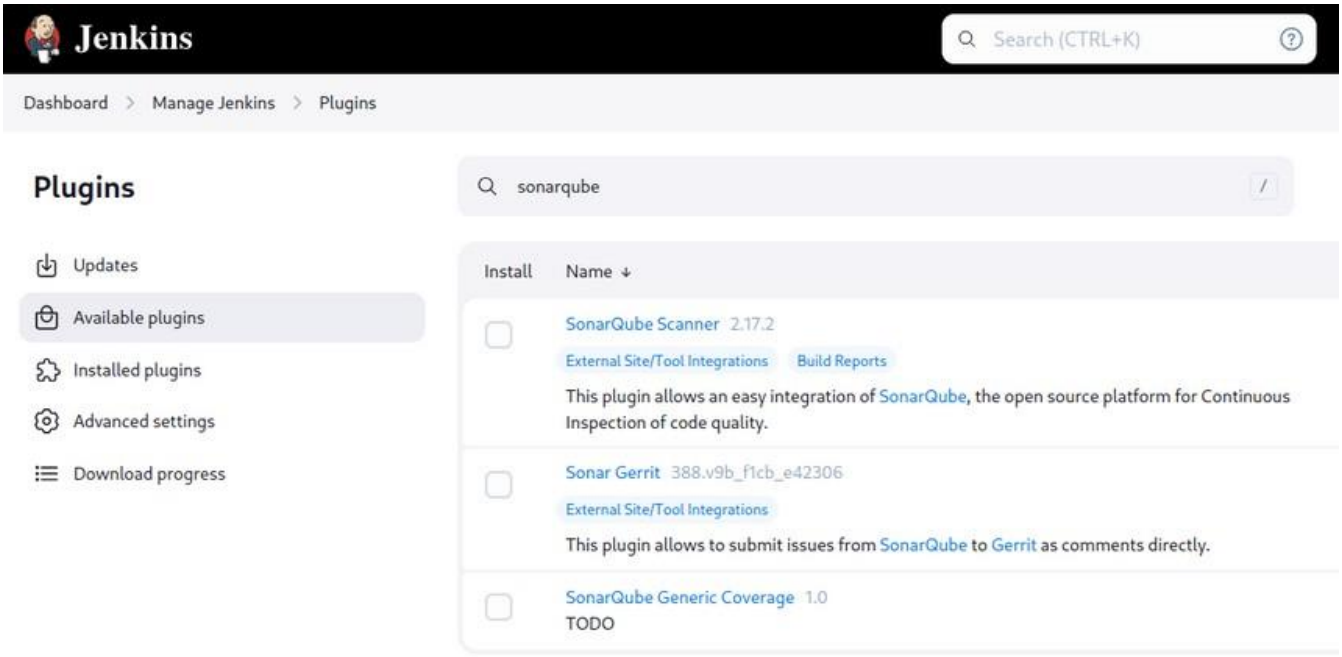
STEPS:
1. First you need to go with the installation part.
› docker run -d --name sonarqube -e SONAR_ES_BOOTSTRAP_CHECKS_DISABLE=true -p 9000:9000 sonarqube:latest
I have used docker, zip file also works well.

2. Then access it on localhost:9000

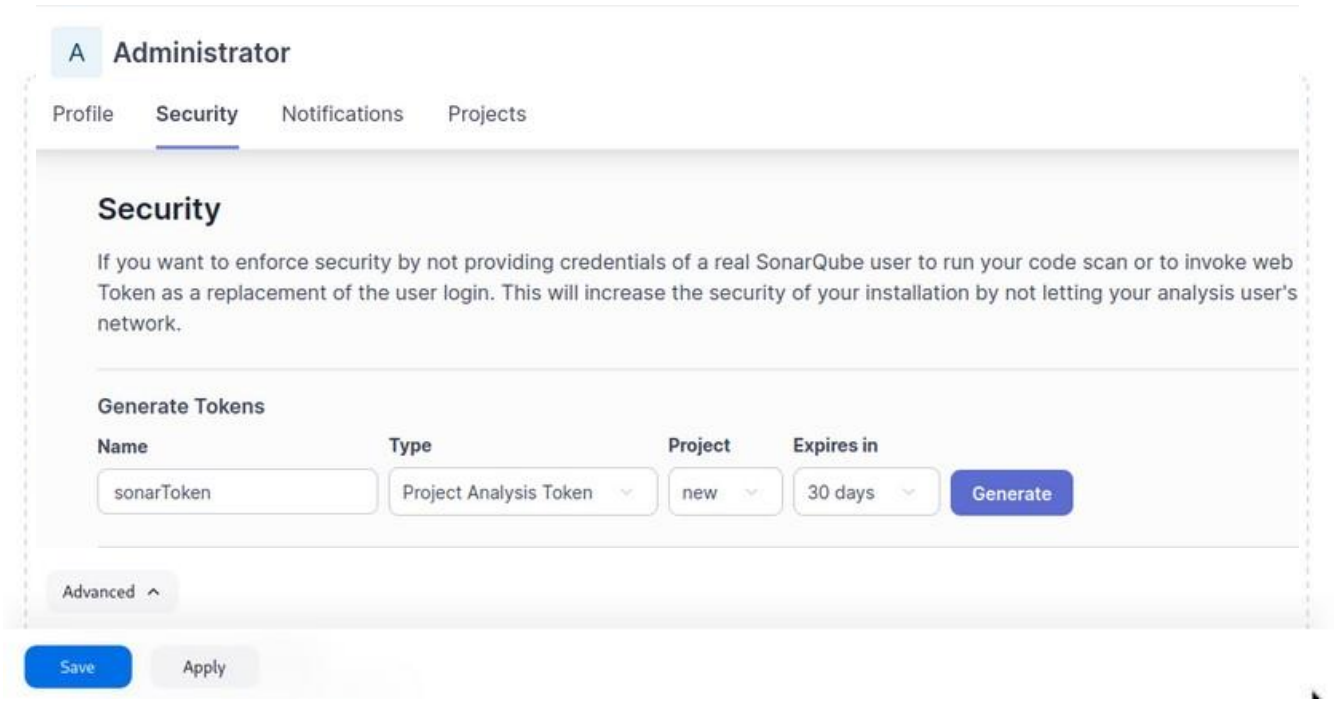




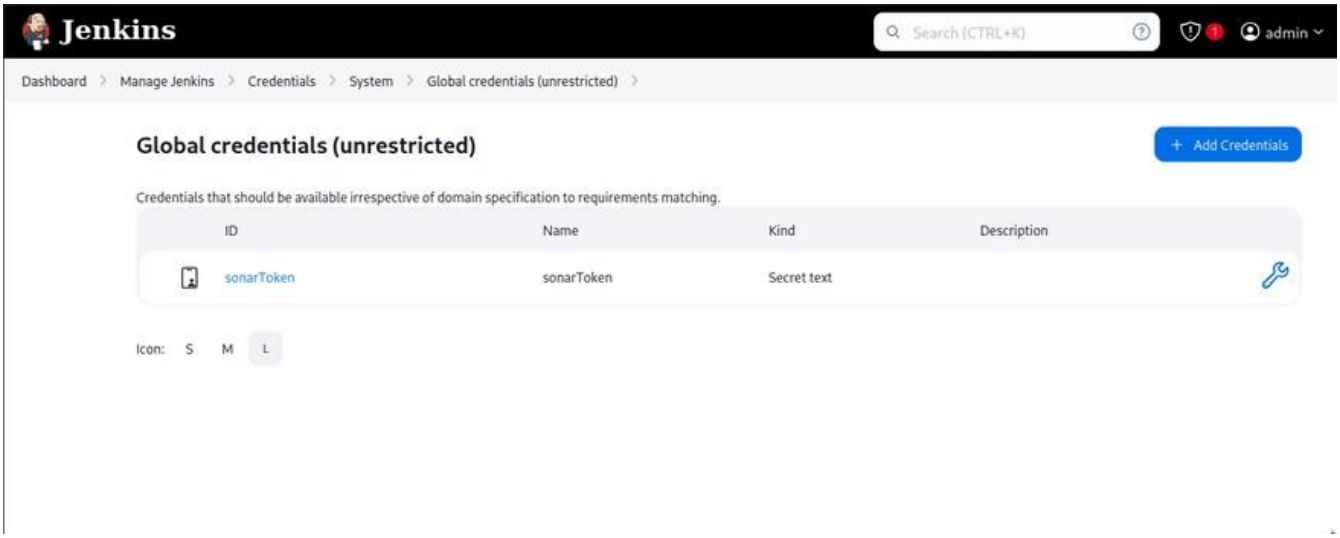
3. Now login to your jenkins dashboard and install SonarQube Scanner plugin.



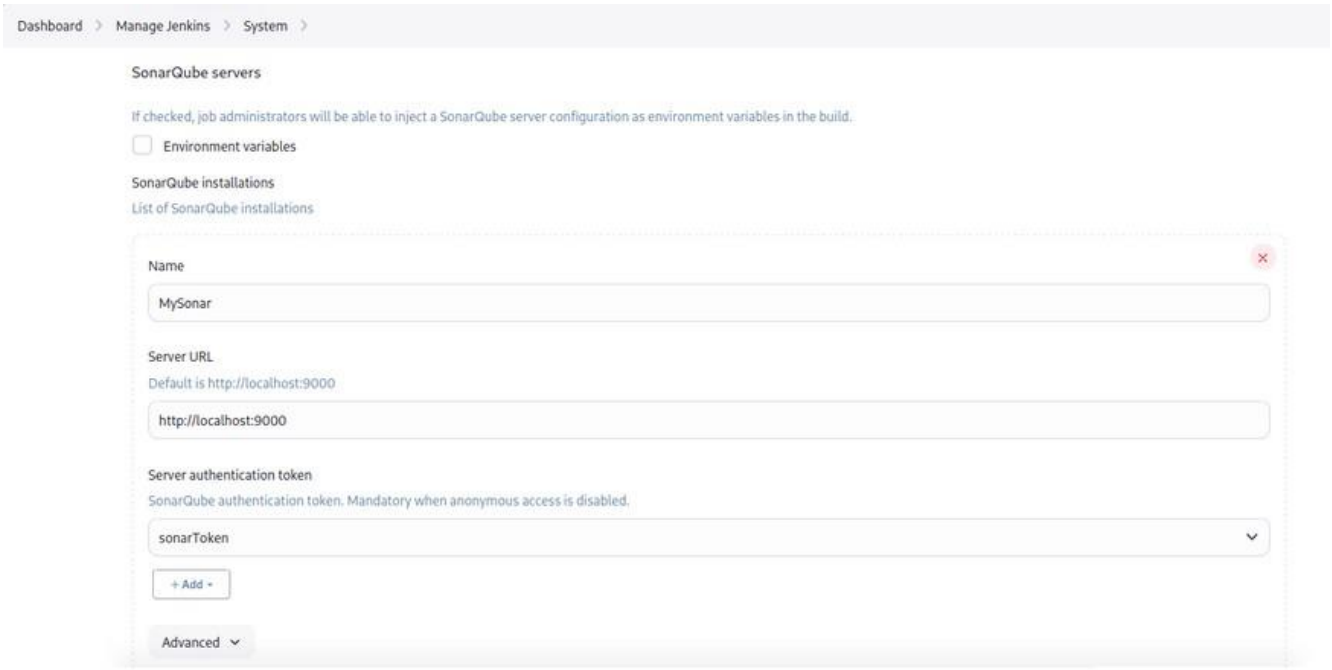
4. Then Go to SonarQube dashboard → Account settings → Secuity → Generate Token. (Make sure you have an project, if not then create)



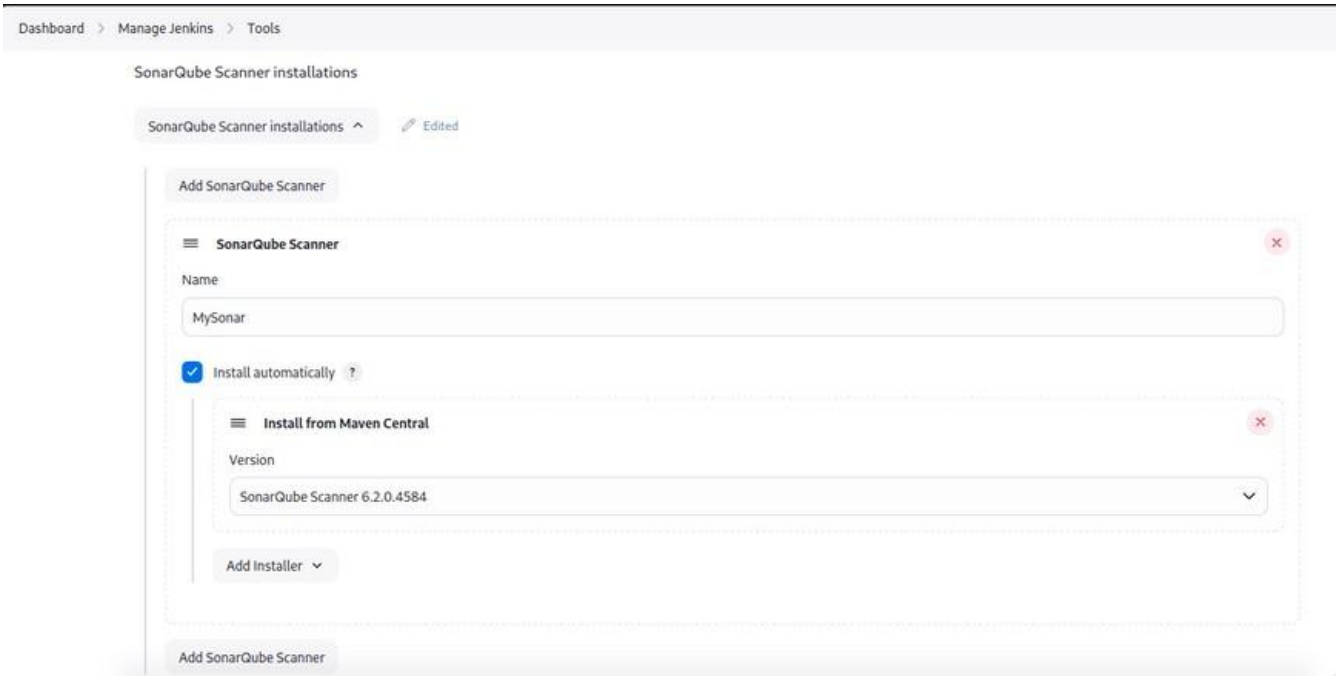
5. Once token is generated add it to Jenkins Credentials as a secret. (Global)



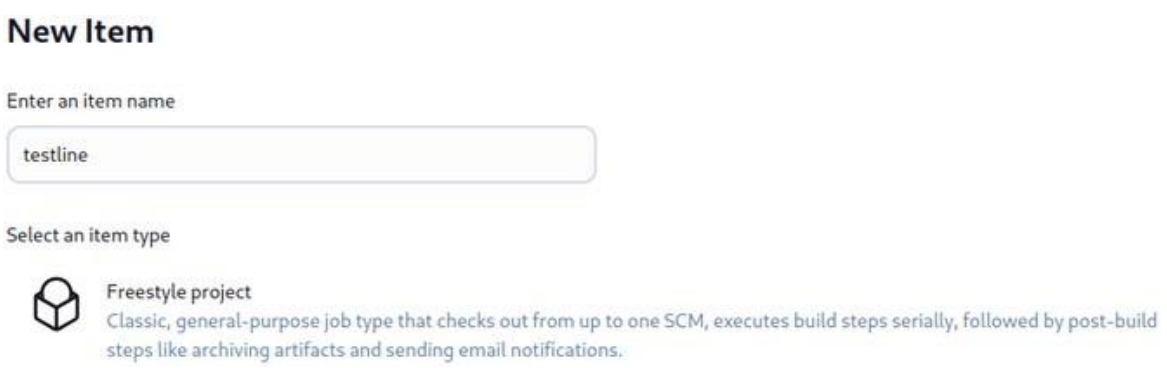
6. Then, Add your SonarQube server with jenkins, and add that credential here and save.



7. Now, setup the SonarQube scanner installations, If you want you can go with manual configuration or just tick "Install Automatically".



8. Start new item, with type "Freestyle project".

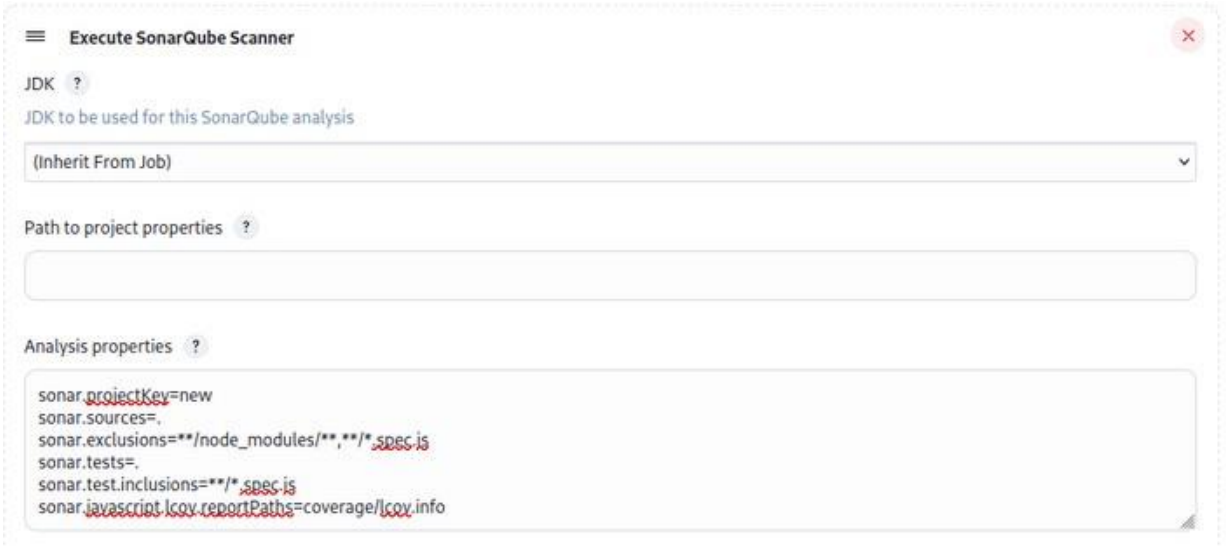


9. Select your source code repository if it on any git server.

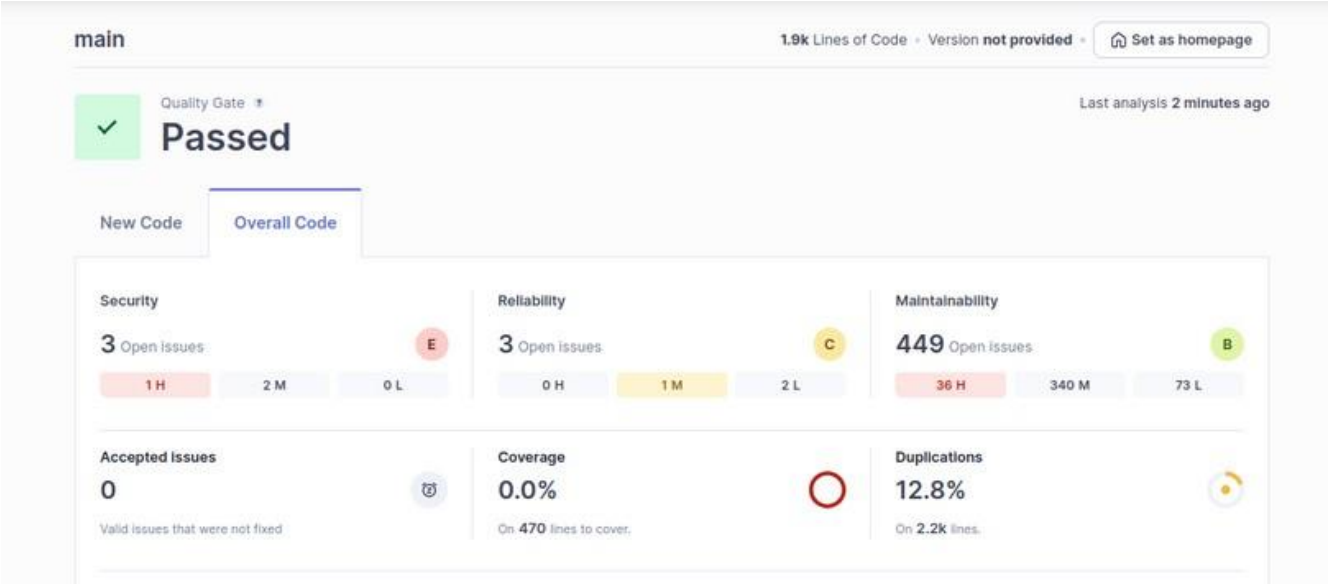
10. You need to create new file 'sonar-project.properties' with following configuration:

```
sonar.projectKey=new
sonar.sources=.
sonar.exclusions=**/node_modules--*,**/*.spec.js
sonar.tests=.
sonar.test.inclusions=**/*.spec.js
sonar.javascript.lcov.reportPaths=coverage/lcov.info
```

Or just add it in Build step.



11. Build the project, then go to sonarQube project.



Conclusion: Thus, we have successfully understood Static Analysis SAST process and learned to integrate Jenkins SAST to SonarQube/GitLab.

EXPERIMENT - 8

Aim: Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static Analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.

Theory:

Security in CI/CD: Integrating security into CI/CD pipelines is crucial in today's software development landscape. This approach, often called "shifting left," involves incorporating security practices earlier in the development process. Static analysis plays a key role by automatically identifying potential security vulnerabilities during the build phase. This early detection allows developers to address security issues before they reach production, significantly reducing the risk and cost associated with security breaches.

Code Quality Metrics: Static analysis tools evaluate code based on various metrics that indicate code quality, maintainability, and potential issues. Common metrics include:

- Code Duplication: Identifies repeated code segments. Excessive duplication can lead to maintenance difficulties and inconsistencies.
- Comment Density: Measures the ratio of comments to code. While not definitive, it can indicate code clarity.
- Code Coverage: Although typically a dynamic analysis metric, it's often reported alongside static analysis results to show the extent of code exercised by tests.

Benefits of Automated Static Analysis in CI/CD:

a) Early Detection: Issues are identified as soon as code is committed, allowing for immediate correction.

b) Consistency: Applies the same quality and security standards across the entire codebase and development team.

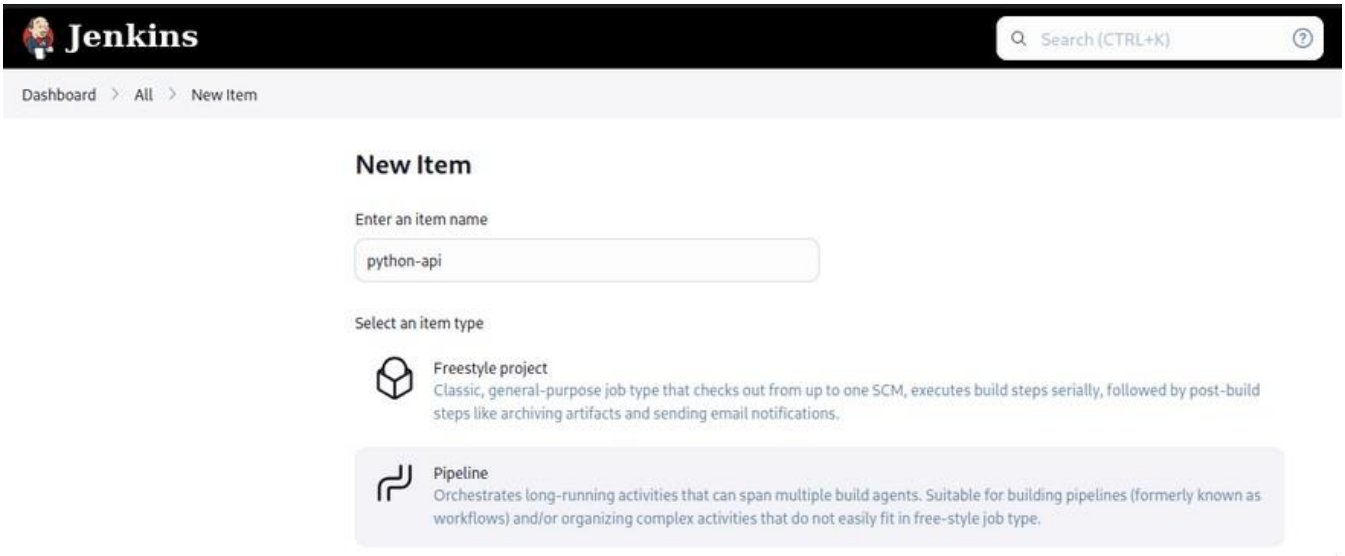
c) Continuous Feedback: Developers receive ongoing insights about their code quality, fostering a culture of continuous improvement.

d) Reduced Review Effort: Automates part of the code review process, allowing human reviewers to focus on higher-level concerns.

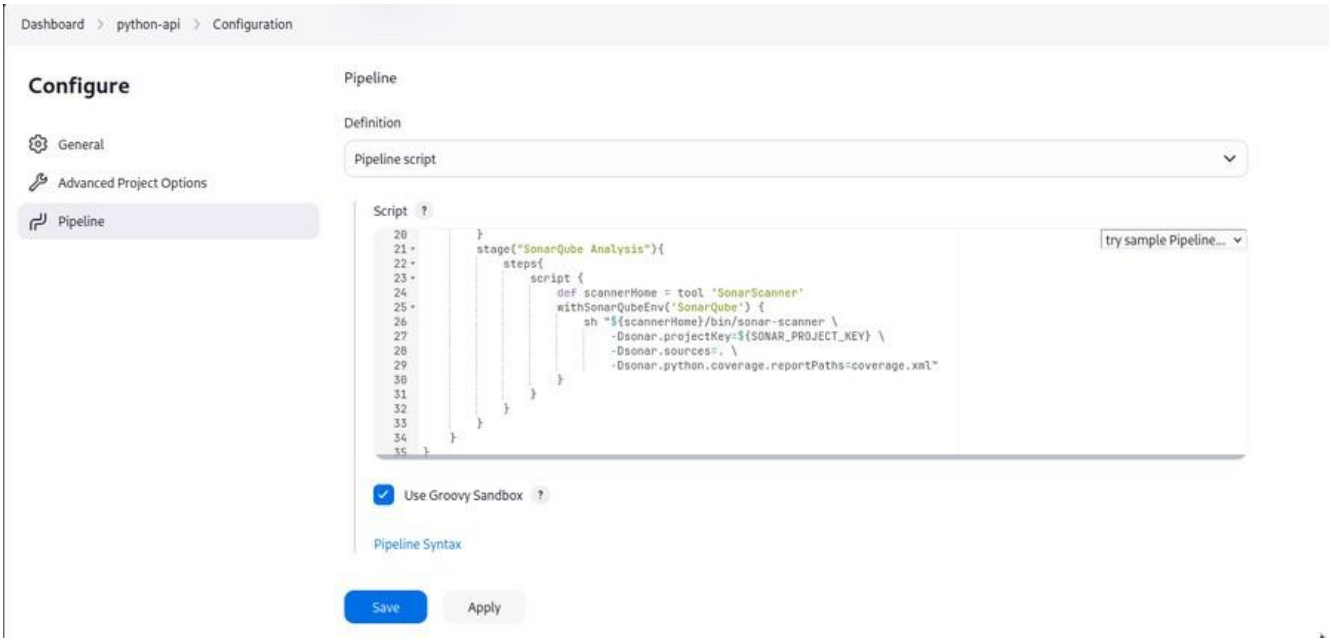
STEPS:

1. Add SonarQube to your jenkins, and setup it.

2. Create a new item in jenkins with ”pipeline” type”



3. Then, Create Pipeline Script in groovy includes steps to build project and run sonar-scanner on it to check for security issues.



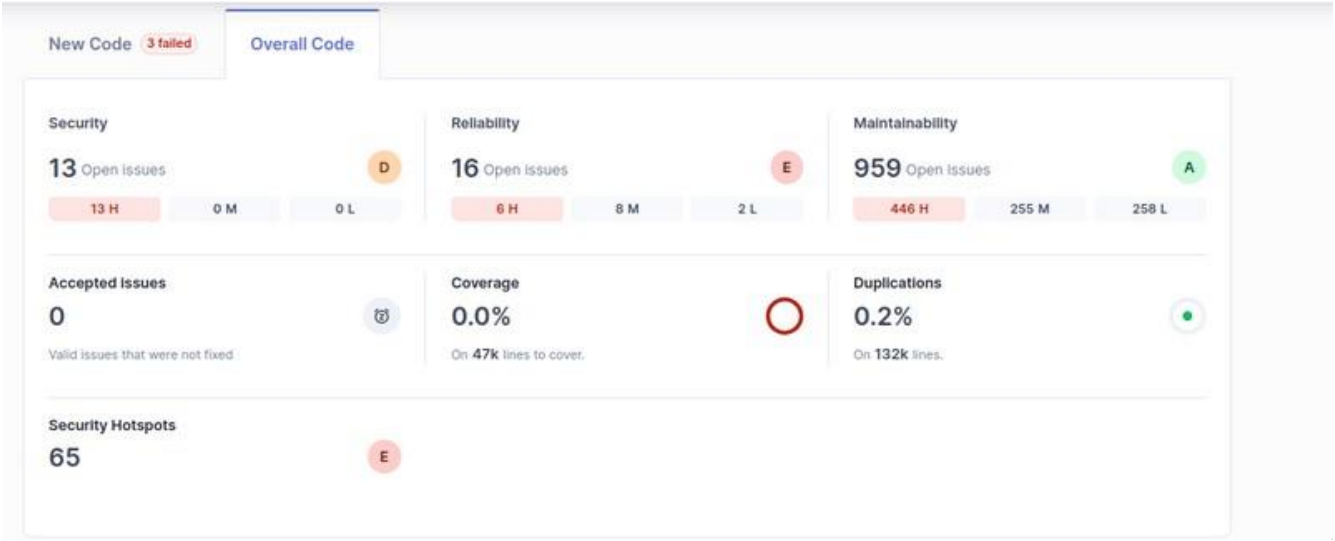
```
pipeline {
    agent any
    environment {
        SONAR_PROJECT_KEY = 'new'
    }
    stages {
        stage("Code"){
            steps{
                git url: "https://github.com/Dark-Kernel/minipy.git", branch: "master"
            }
        }
        stage("Install dependencies") {
            steps{
                sh "python -m venv env"
                sh "source env/bin/activate"
                sh "pip install -r requirements.txt"
            }
        }
        stage("SonarQube Analysis"){
            steps{
                script {
                    def scannerHome = tool 'SonarScanner'
                    withSonarQubeEnv('SonarQube') {
                        sh "${scannerHome}/bin/sonar-scanner \
                            -Dsonar.projectKey=${SONAR_PROJECT_KEY} \
                            -Dsonar.sources=. \
                            -Dsonar.python.coverage.reportPaths=coverage.xml"
                    }
                }
            }
        }
    }
}
```

4. Then save, and build.

```
17:42:00.898 INFO  Analysis report uploaded in 196ms
17:42:00.899 INFO  ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=new
17:42:00.901 INFO  Note that you will be able to access the updated dashboard once the server has processed the submitted
analysis report
17:42:00.901 INFO  More about the report processing at http://localhost:9000/api/ce/task?id=be529038-9465-42b4-
a841-880063f99958
17:42:01.005 INFO  Analysis total time: 49.249 s
17:42:01.006 INFO  SonarScanner Engine completed successfully
17:42:01.202 INFO  EXECUTION SUCCESS
17:42:01.203 INFO  Total time: 57.455s
[Pipeline] }
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

5. Then On Successful build you will have your report of sonar-scanner containing Failed checks and vulnerabilities.

Visit SonarQube Dashboard and fix your application security issues.



Conclusion: Thus, we have successfully Created a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static Analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.

EXPERIMENT - 9

Aim: To Understand Continuous monitoring and Installation and configuration of Nagios Core, Nagios Plugins and NRPE (Nagios Remote Plugin Executor) on Linux Machine.

Theory:

Nagios is an event monitoring system that offers monitoring and alerting services for servers, switches, applications and services. It alerts users when things go wrong and alerts them a second time when the problem has been resolved.

Extensively monitor critical components, applications, and systems with our own add-ons and thousands of third-party add-ons for comprehensive coverage.

Ensure optimal server performance for all your monitoring endeavors with the Nagios Core 4 monitoring engine, which utilizes high-efficiency processes for scalability and effectiveness.

Some key features of Nagios Core:

Visibility:

Enhance user access to relevant information with a centralized view of the monitoring data and third-party insights that are most critical to you.

Customizability:

Give users and team members the flexibility to tailor layout, design, and preferences on a per-user basis with a customizable GUI.

Multi-Tenant Capabilities and Ease of Use:

Simplify administration with advanced user management to efficiently manage user accounts and ensure clients only see the infrastructure components they're authorized for.

Nagios NRPE: (Nagios Remote Plugin Executor) is an agent used by Nagios XI for communicating with remote hosts. It allows you to run Nagios plugins on remote machines, enabling the monitoring of remote machine metrics such as disk usage, CPU load, and more.

Nagios Plugins are external programs or scripts that run from the command line to check the status of hosts and services on a network. They are an essential part of the Nagios monitoring system, as they provide the necessary information for Nagios Core to determine the current status of monitored resources.

STEPS to install Nagios Core:

1. Install dependencies

~$ sudo apt install build-essential apache2 php libapache2-mod-php php-gd libgd-dev unzip automake

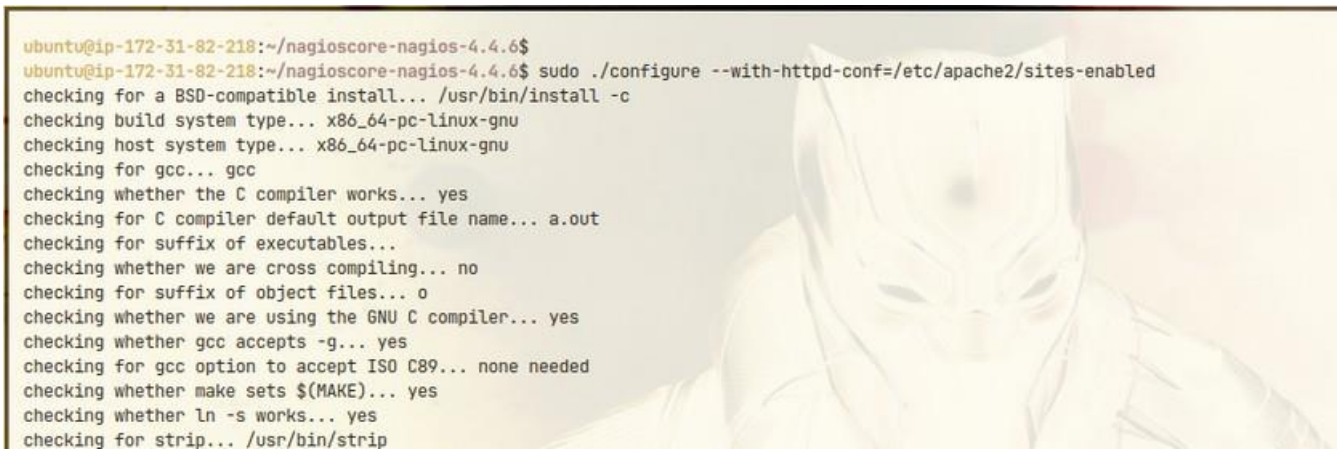
2. Download the latest version archive file.

~$ wget https://github.com/NagiosEnterprises/nagioscore/archive/nagios-4.4.6.tar.gz

~$ tar xzf nagios-4.4.6.tar.gz

~$ cd nagioscore-nagios-4.4.6/

3. Configure it.



4. Start Compilation.

~$ sudo make all

```
ubuntu@ip-172-31-82-218:~/nagioscore-nagios-4.4.6$ sudo make all
cd ./base && make
make[1]: Entering directory '/home/ubuntu/nagioscore-nagios-4.4.6/base'
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o nagios.o nagios.c
nagios.c: In function 'main':
nagios.c:611:25: warning: ignoring return value of 'asprintf' declared with attribute 'warn_unused_result' [-Wunused-result]
  611 |                         asprintf(&mac->x[MACRO_PROCESSSTARTTIME], "%llu", (unsigned long long)program_start);
      |                         ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
nagios.c:841:25: warning: ignoring return value of 'asprintf' declared with attribute 'warn_unused_result' [-Wunused-result]
  841 |                         asprintf(&mac->x[MACRO_EVENTSTARTTIME], "%llu", (unsigned long long)event_start);
      |                         ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
nagios.c: In function 'nagios_core_worker':
nagios.c:176:17: warning: ignoring return value of 'read' declared with attribute 'warn_unused_result' [-Wunused-result]
  176 |                 read(sd, response + 3, sizeof(response) - 4);
      |                 ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
nagios.c: In function 'test_path_access':
nagios.c:122:17: warning: ignoring return value of 'asprintf' declared with attribute 'warn_unused_result' [-Wunused-result]
  122 |                 asprintf(&path, "%s/%s", p, program);
      |                 ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o broker.o broker.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o nebmods.o nebmods.c
```

5. Add user of nagios

~$ sudo useradd nagios

~$ sudo usermod -a -G nagios www-data

6. Make install

~$ sudo make install

```
ubuntu@ip-172-31-82-218:~/nagioscore-nagios-4.4.6$
ubuntu@ip-172-31-82-218:~/nagioscore-nagios-4.4.6$ sudo make install
cd ./base && make install
make[1]: Entering directory '/home/ubuntu/nagioscore-nagios-4.4.6/base'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/usr/bin/install -c -s -m 774 -o nagios -g nagios nagiostats /usr/local/nagios/bin
make[1]: Leaving directory '/home/ubuntu/nagioscore-nagios-4.4.6/base'
cd ./cgi && make install
make[1]: Entering directory '/home/ubuntu/nagioscore-nagios-4.4.6/cgi'
make install-basic
make[2]: Entering directory '/home/ubuntu/nagioscore-nagios-4.4.6/cgi'
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/sbin
for file in *.cgi; do \
        /usr/bin/install -c -s -m 775 -o nagios -g nagios $file /usr/local/nagios/sbin; \
done
make[2]: Leaving directory '/home/ubuntu/nagioscore-nagios-4.4.6/cgi'
make[1]: Leaving directory '/home/ubuntu/nagioscore-nagios-4.4.6/cgi'
cd ./html && make install
```

7. compile other required modules

~$ sudo make install-daemoninit install-commandmode install-config install-webconf

```
ubuntu@ip-172-31-82-218:~/nagioscore-nagios-4.4.6$
ubuntu@ip-172-31-82-218:~/nagioscore-nagios-4.4.6$ sudo make install-daemoninit install-commandmode install-config install-webconf
/usr/bin/install -c -m 755 -d -o root -g root /lib/systemd/system
/usr/bin/install -c -m 755 -o root -g root startup/default-service /lib/systemd/system/nagios.service
Created symlink /etc/systemd/system/multi-user.target.wants/nagios.service → /lib/systemd/system/nagios.service.

*** Init script installed ***

/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/var/rw
chmod g+s /usr/local/nagios/var/rw

*** External command directory configured ***

/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc/objects
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/nagios.cfg /usr/local/nagios/etc/nagios.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/cgi.cfg /usr/local/nagios/etc/cgi.cfg
/usr/bin/install -c -b -m 660 -o nagios -g nagios sample-config/resource.cfg /usr/local/nagios/etc/resource.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/templates.cfg /usr/local/nagios/etc/objects/templates.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/commands.cfg /usr/local/nagios/etc/objects/commands.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/contacts.cfg /usr/local/nagios/etc/objects/contacts.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/timeperiods.cfg /usr/local/nagios/etc/objects/timeperiods.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/localhost.cfg /usr/local/nagios/etc/objects/localhost.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/windows.cfg /usr/local/nagios/etc/objects/windows.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/printer.cfg /usr/local/nagios/etc/objects/printer.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/switch.cfg /usr/local/nagios/etc/objects/switch.cfg

*** Config files installed ***
```

STEPS to install Nagios Plugin:

1. Now compile and install Nagios Plugins

~$ wget https://github.com/nagios-plugins/nagios-plugins/archive/release-2.3.3.tar.gz
~$ tar xzf release-2.3.3.tar.gz
~$ cd nagios-plugins-release-2.3.3/

2. Setup, Configure, compile and install.

~$ sudo ./tools/setup

```
ubuntu@ip-172-31-82-218:~/nagios-plugins-release-2.3.3$ sudo ./tools/setup
Found GNU Make at /usr/bin/gmake ... good.
configure.ac:46: installing 'build-aux/compile'
configure.ac:12: installing 'build-aux/config.guess'
configure.ac:12: installing 'build-aux/config.sub'
configure.ac:9: installing 'build-aux/install-sh'
configure.ac:9: installing 'build-aux/missing'
Makefile.am: installing './INSTALL'
gl/Makefile.am: installing 'build-aux/depcomp'
parallel-tests: installing 'build-aux/test-driver'
configure.ac:47: warning: The macro `AC_GNU_SOURCE' is obsolete.
configure.ac:47: You should run autoupdate.
./lib/autoconf/specific.m4:312: AC_GNU_SOURCE is expanded from...
gl/m4/gnulib-comp.m4:34: gl_EARLY is expanded from...
configure.ac:47: the top level
configure.ac:47: warning: The macro `AC_HELP_STRING' is obsolete.
configure.ac:47: You should run autoupdate.
./lib/autoconf/general.m4:204: AC_HELP_STRING is expanded from...
```

~$ sudo ./configure

```
ubuntu@ip-172-31-82-218:~/nagios-plugins-release-2.3.3$ sudo ./configure
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a race-free mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking whether to enable maintainer-specific portions of Makefiles... yes
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
```

~$ sudo make

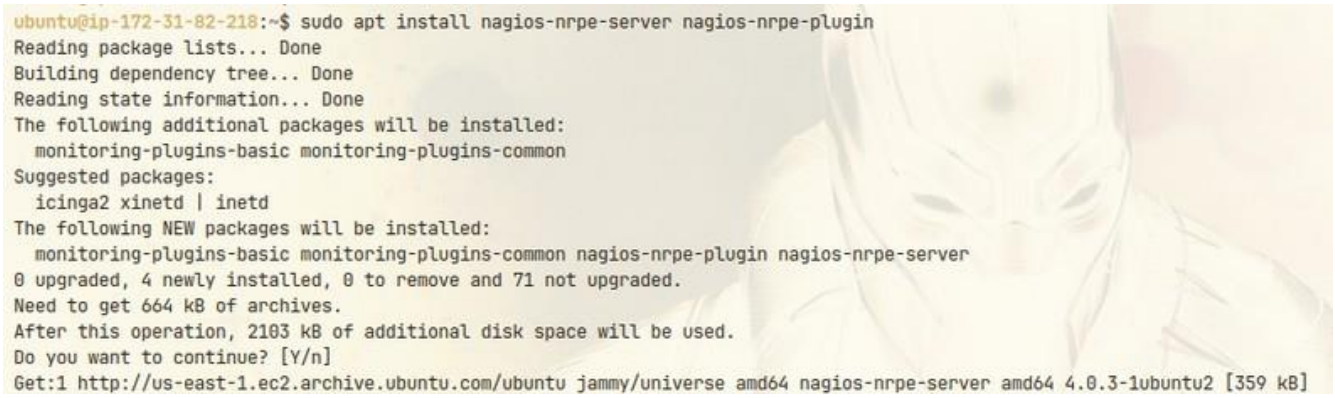
```
ubuntu@ip-172-31-82-218:~/nagios-plugins-release-2.3.3$ sudo make
make  all-recursive
make[1]: Entering directory '/home/ubuntu/nagios-plugins-release-2.3.3'
Making all in gl
make[2]: Entering directory '/home/ubuntu/nagios-plugins-release-2.3.3/gl'
rm -f alloca.h-t alloca.h && \
{ echo '/* DO NOT EDIT! GENERATED AUTOMATICALLY! */'; \
  cat ./alloca.in.h; \
} > alloca.h-t && \
mv -f alloca.h-t alloca.h
rm -f c++defs.h-t c++defs.h && \
sed -n -e '/_GL_CXXDEFS/,$p' \
  < ../build-aux/snippet/c++defs.h \
  > c++defs.h-t && \
mv c++defs.h-t c++defs.h
rm -f warn-on-use.h-t warn-on-use.h && \
sed -n -e '/^.ifndef/,$p' \
  < ../build-aux/snippet/warn-on-use.h \
  > warn-on-use.h-t && \
```

~$ sudo make install

```
ubuntu@ip-172-31-82-218:~/nagios-plugins-release-2.3.3$ sudo make install
Making install in gl
make[1]: Entering directory '/home/ubuntu/nagios-plugins-release-2.3.3/gl'
make  install-recursive
make[2]: Entering directory '/home/ubuntu/nagios-plugins-release-2.3.3/gl'
make[3]: Entering directory '/home/ubuntu/nagios-plugins-release-2.3.3/gl'
make[4]: Entering directory '/home/ubuntu/nagios-plugins-release-2.3.3/gl'
if test yes = no; then \
  case 'linux-gnu' in \
    darwin[56]*) \
      need_charset_alias=true ;; \
    darwin* | cygwin* | mingw* | pw32* | cegcc*) \
      need_charset_alias=false ;; \
    *) \
      need_charset_alias=true ;; \
  esac ; \
else \
  need_charset_alias=false ; \
```

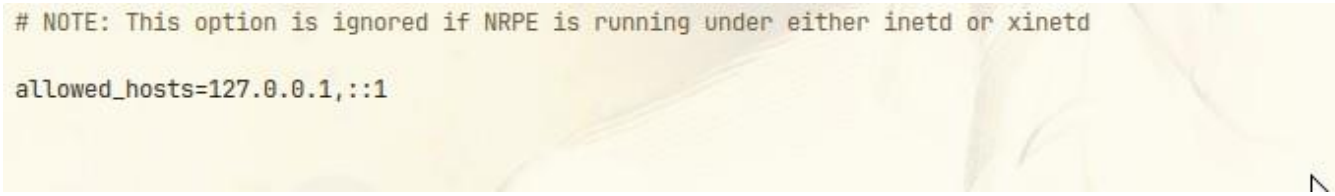
Now install NRPE

~$ sudo apt install nagios-nrpe-server nagios-nrpe-plugin



Now, it's time to configure.

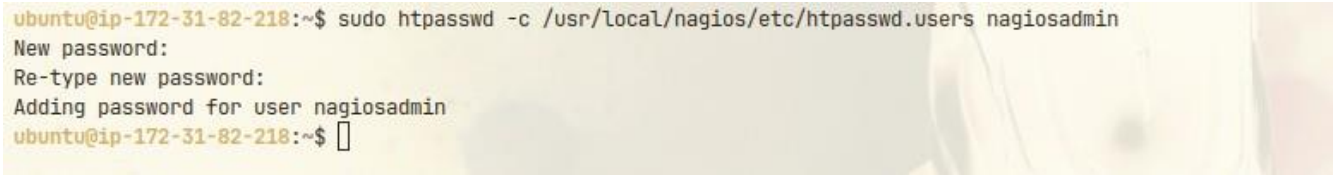
~$ sudo nano /etc/nagios/nrpe.cfg



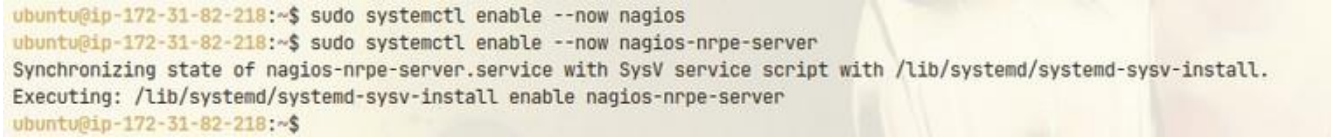
Updated it with your Nagios server IP.

Setup apache password.

~$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin



Enable & Start Nagios Service.
~$ sudo systemctl enable /-now nagios
~$ sudo systemctl enable /-now nagios-nrpe-server



You can now access it on your server's IP at '/nagios'.
Enter your username & password.

And, After logging in you will the home page of Nagios.

Conclusion: Thus, we have successfully Understood Continuous monitoring and Installation and configuration of Nagios Core, Nagios Plugins and NRPE (Nagios Remote Plugin Executor) on Linux Machine.

EXPERIMENT - 10

Aim: To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Node js.

Theory:

AWS Lambda:

AWS Lambda is a serverless compute service provided by Amazon Web Services (AWS). It allows developers to run code in response to events without provisioning or managing servers. This event-driven, serverless Function as a Service (FaaS) enables rapid and cost-effective modern applications development.

Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, and logging. With Lambda, all you need to do is supply your code in one of the language runtimes that Lambda supports.

You organize your code into Lambda functions. The Lambda service runs your function only when needed and scales automatically. You only pay for the compute time that you consume—there is no charge when your code is not running.

Workflow:

- Function: Your code that Lambda runs
- Event Source: AWS service or custom application that triggers your function
- Trigger: Association between an event source and a Lambda function
- Execution Environment: Secure and isolated runtime environment for your Lambda function

Some examples of various AWS Lambda functions:

1. Image Resizing: A Lambda function can be triggered by an Amazon S3 object upload and resize images to different dimensions, making them suitable for different use cases (e.g., thumbnails, banners).

2. Real-time Data Processing: Lambda functions can be used to process real-time data from IoT devices, such as sensor readings, and trigger actions like sending notifications or updating databases.

3. API Gateway Integration: Lambda functions can be used as the backend for API Gateway, handling API requests and responses, and integrating with other AWS services like DynamoDB or S3.

4. Chatbot or Virtual Assistant: A Lambda function can be triggered by voice or text input and respond with relevant information or actions, integrating with services like Amazon Lex or Amazon Comprehend.

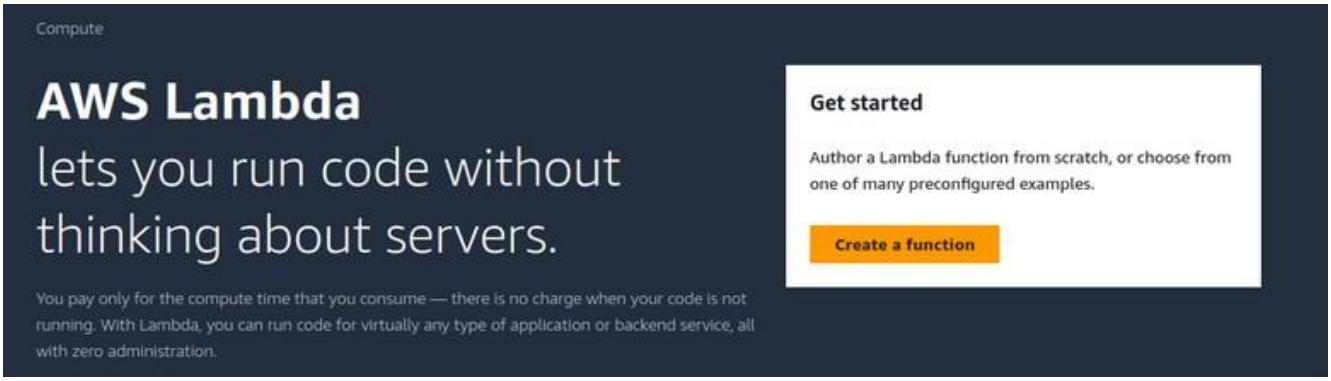
5. Email Processing: Lambda functions can be used to process and analyze email attachments, such as extracting metadata or sending notifications to team members.

6. Log Processing: Lambda functions can be triggered by Amazon CloudWatch Logs and process log data, such as aggregating metrics, detecting anomalies, or sending alerts.

7. Webhook Processing: Lambda functions can be used to process webhooks from third-party services, such as payment gateways or social media platforms, and trigger actions in your application.

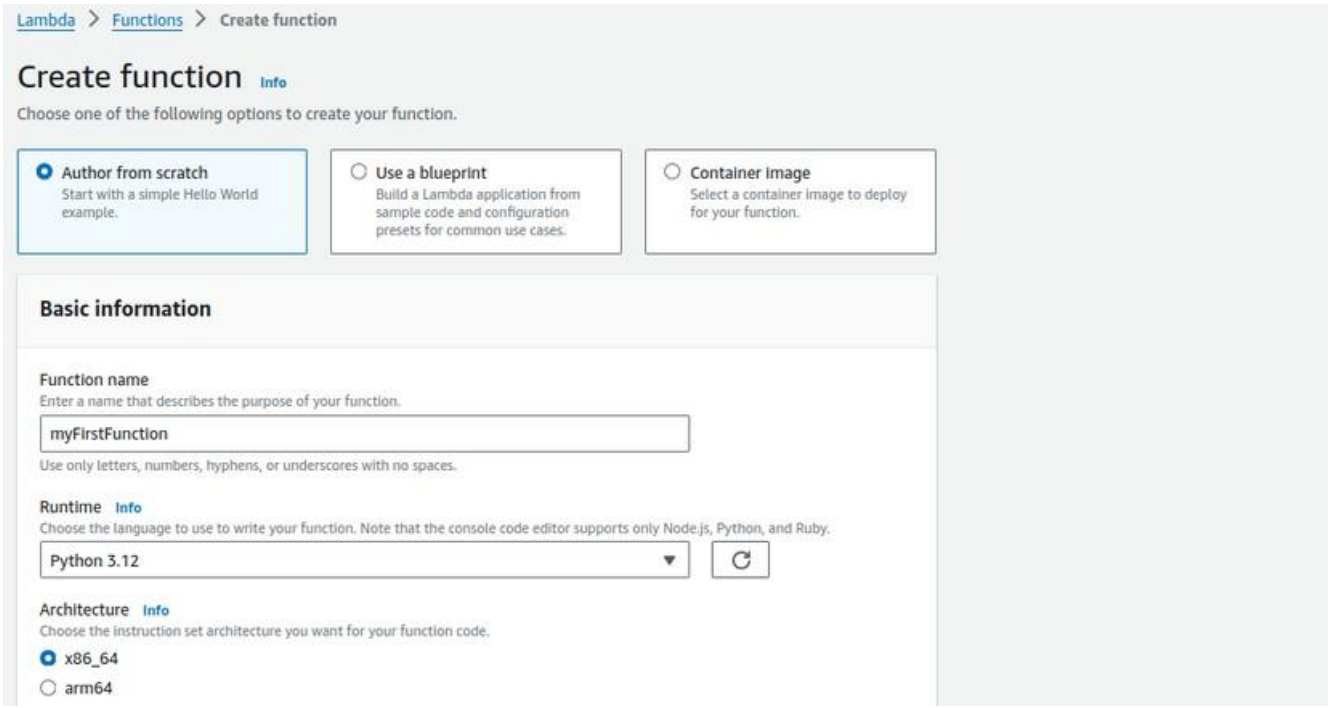
STEPS:

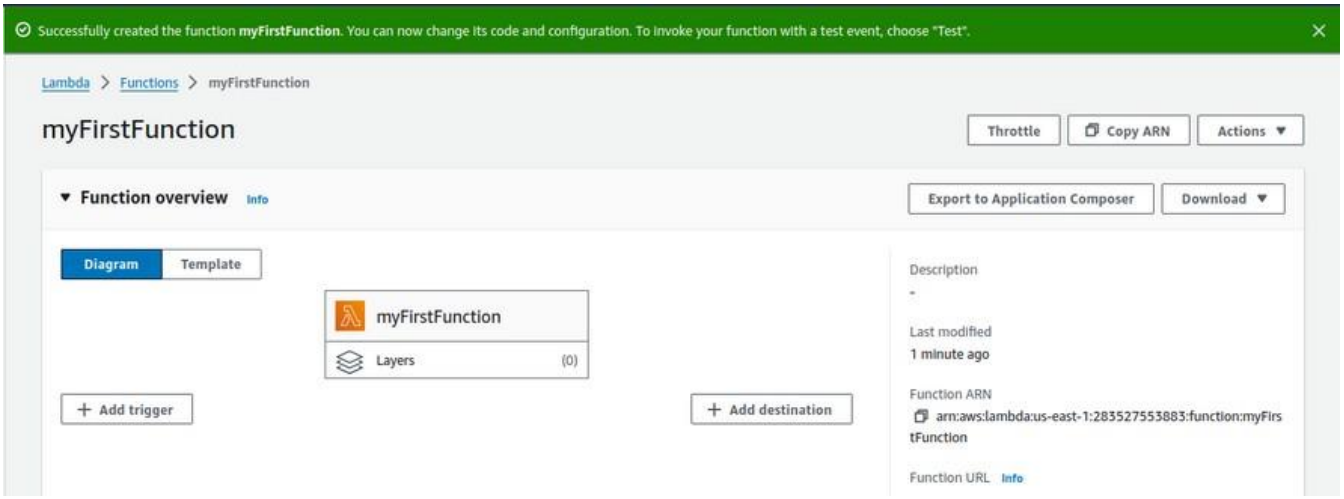
1. Sign in to the AWS Management Console, and search for Aws lambda service.



2. Click create a function

Enter the details as given below. → Create





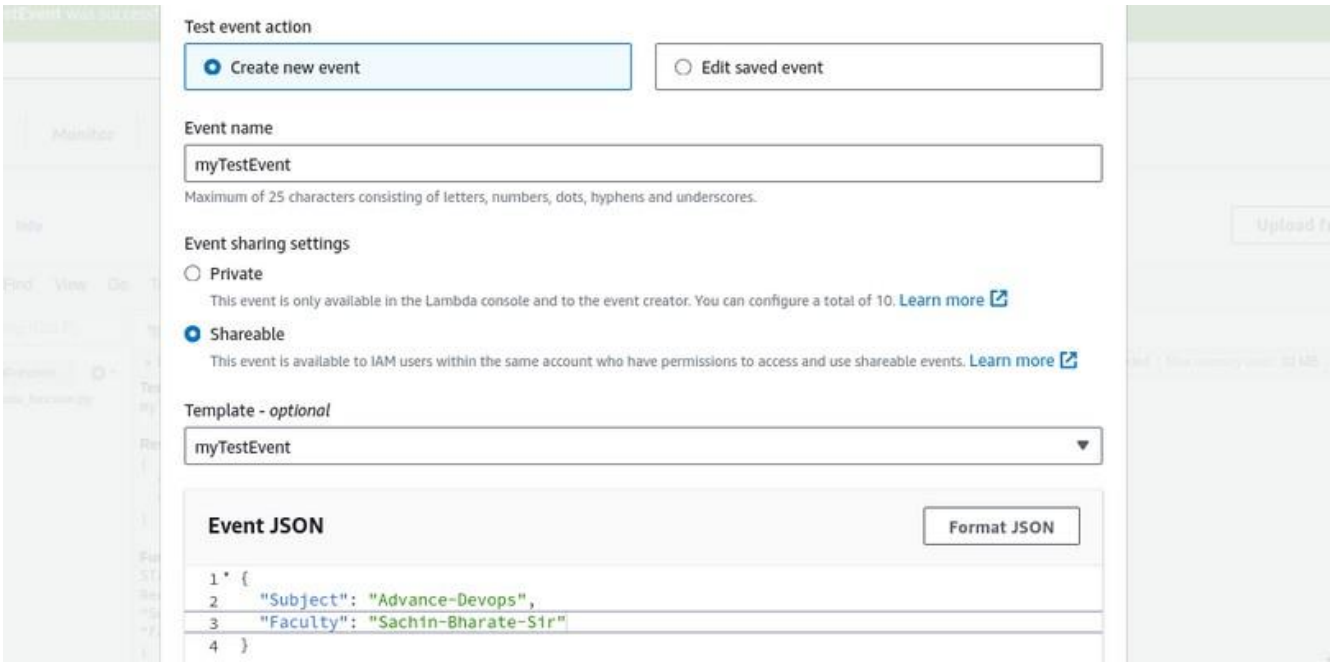
Scroll down to code section,

Now change the default code with this:

```
import json
def lambda_handler(event, context):


    print("Received event: " + json.dumps(event, indent=2))
    message = 'Hello from Lambda!'
    return {
        'statusCode': 200,
        'body': json.dumps(message)
    }
```

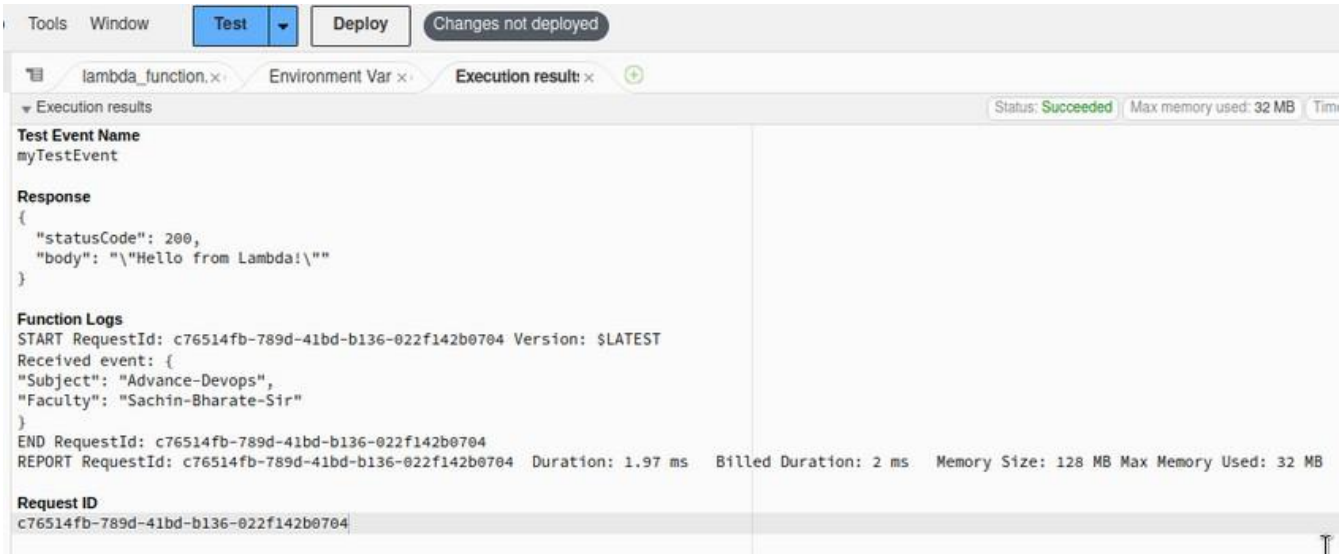
Then, click on deploy, After it is deployed click on "Test"



Give name to event, You can change event json data accordingly, then click save.

Then again click on test and you will see the output of it.

Note: You can create multiple events for testing purpose.



The output contains that json and it is exact output of our code.

Conclusion: Thus, we have successfully understood AWS Lambda, its workflow, various functions and created our first Lambda functions using Python / Java / Node js.

Experiment No: 8

Aim: Study the use of network reconnaissance tools like WHOIS, dig, traceroute, nslookup to gather information about networks and domain registrars.

Theory:

WHOIS :

WHOIS is the Linux utility for searching an object in a WHOIS database. The WHOIS database of a domain is the publicly displayed information about a domains ownership, billing, technical, administrative, and name server information. Running a WHOIS on your domain will look the domain up at the registrar for the domain information. All domains have WHOIS information. WHOIS database can be queried to obtain the following information via WHOIS:

- Administrative contact details, including names, email addresses, and telephone numbers
- Mailing addresses for office locations relating to the target organization
- Details of authoritative name servers for each given domain

Example: Querying Facebook.com

ssc@ssc-OptiPlex-380:~$ whois facebook.com **Whois Server Version 2.0** Domain names in the .com and .net domains can now be registered with many different competing registrars. Go to http://www.internic.net for detailed information. Server Name: FACEBOOK.COM.BRETLANDTRUSTMERCHANDISINGDEPART.COM

IP Address: 69.63.176.11

Registrar: GOOGLE INC.

Whois Server: whois.rrpproxy.net

Referral URL: http://domains.google.com

Server Name: FACEBOOK.COM.DISABLE.YOUR.TIMELINE.NOW.WITH.THE.ORIGINAL.TIMELINE-REMOVE.NET

IP Address: 8.8.8.8

Registrar: ENOM, INC.

Whois Server: whois.enom.com

Referral URL: http://www.enom.com

Server Name: FACEBOOK.COM.GET.ONE.MILLION.DOLLARS.AT.WWW.UNIMUNDI.COM

IP Address: 209.126.190.70

Registrar: PDR LTD. D/B/A PUBLICDOMAINREGISTRY.COM

Whois Server: whois.PublicDomainRegistry.com

Referral URL: http://www.PublicDomainRegistry.com

Dig - Dig is a networking tool that can query DNS servers for information. It can be very helpful for diagnosing problems with domain pointing and is a good way to verify that your configuration is working. The most basic way to use dig is to specify the domain we wish to query:

dig example.com

$ dig example.com

Traceroute - traceroute prints the route that packets take to a network host. Traceroute utility uses the TTL field in the IP header to achieve its operation. For users who are new to TTL field, this field describes how much hops a particular packet will take while traveling on network. So, this effectively outlines the lifetime of the packet on network. This field is usually set to 32 or 64. Each time the packet is held on an intermediate router, it decreases the TTL value by 1. When a router finds the TTL value of 1 in a received packet then that packet is not forwarded but instead discarded. After discarding the packet, router sends an ICMP error message of ―Time exceeded‖ back to the source from where packet generated. The ICMP packet that is sent back contains the IP address of the router. So now it can be easily understood that traceroute operates by sending packets with TTL value starting from 1 and then incrementing by one each time. Each time a router receives the packet, it checks the TTL field, if TTL field is 1 then it discards the packet and sends the ICMP error packet containing its IP address and this is what traceroute requires. So traceroute incrementally fetches the IP of all the routers between the source and the destination.

Example: traceroute example.com

Nslookup - The nslookup command is used to query internet name servers interactively for information. nslookup, which stands for "name server lookup", is a useful tool for finding out information about a named domain. By default, nslookup will translate a domain name to an IP address (or vice versa). For instance, to find out what the IP address of microsoft.com is, you could run the command:

$nslookup microsoft.com

Conclusion: Hence we have successfully studied various network reconnaissance tools.

Experiment No: 9

Aim: Study of packet sniffer tools wireshark: - a. Observe performance in promiscuous as well as non-promiscuous mode. b. Show the packets can be traced based on different filters.

Theory:

Wireshark, a network analysis tool formerly known as Ethereal, captures packets in real time and display them in human-readable format. Wireshark includes filters, color-coding and other features that let you dig deep into network traffic and inspect individual packets.

Features of Wireshark :

- Available for UNIX and Windows.
- Capture live packet data from a network interface.
- Open files containing packet data captured with tcpdump/WinDump, Wireshark, and a
- number of other packet capture programs.
- Import packets from text files containing hex dumps of packet data.
- Display packets with very detailed protocol information.
- Export some or all packets in a number of capture file formats.
- Filter packets on many criteria.
- Search for packets on many criteria.
- Colorize packet display based on filters.
- Create various statistics.

Capturing Packets

After downloading and installing wireshark, you can launch it and click the name of an interface under Interface List to start capturing packets on that interface. For example, if you want to capture traffic on the wireless network, click your wireless interface. You can configure advanced features by clicking Capture Options.

Installation of Wireshark:

sudo apt-get install wireshark

After downloading and installing wireshark, you can launch it and click the name of an interface under Interface List to start capturing packets on that interface. Figure 5 shows the example of

interface list in wireshark. For example, if you want to capture traffic on the wireless network, click your wireless interface. You can configure advanced features by clicking Capture Options.

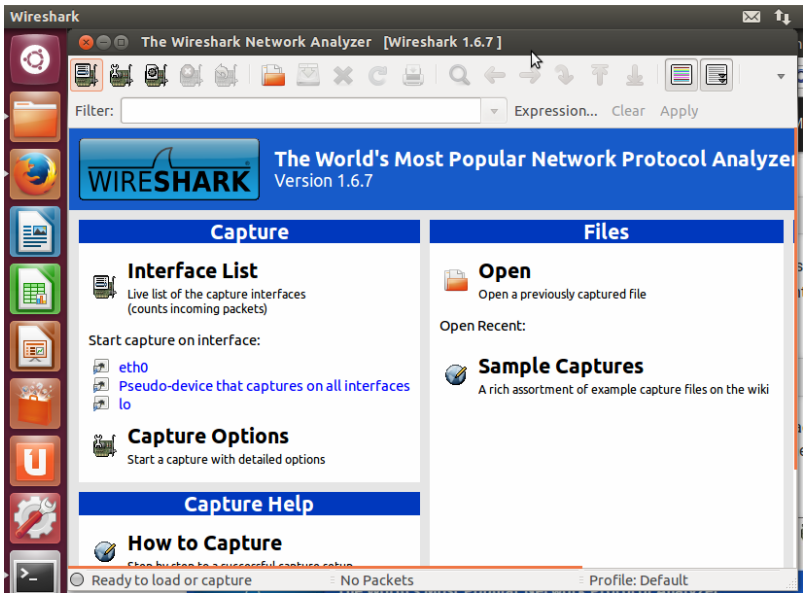


Figure 5: Interface List using Wireshark

As soon as you click the interface‘s name, you‘ll see the packets start to appear in real time. Wireshark captures each packet sent to or from your system. If you‘re capturing on a wireless interface and have promiscuous mode enabled in your capture options, you‘ll also see other the other packets on the network.

Click the stop capture button near the top left corner of the window when you want to stop capturing traffic. Wireshark uses colors to help you identify the types of traffic at a glance. By default, green is TCP traffic, dark blue is DNS traffic, light blue is UDP traffic, and black identifies TCP packets with problems — for example, they could have been delivered out-of-order.

Filtering Packets

If you‘re trying to inspect something specific, such as the traffic a program sends when phoning home, it helps to close down all other applications using the network so you can narrow down the traffic. Still, you‘ll likely have a large amount of packets to sift through. That‘s where filters of Wireshark come in. Figure 6 shows filter in Wireshark and traffic running in Wireshark.

The most basic way to apply a filter is by typing it into the filter box at the top of the window and clicking Apply (or pressing Enter). For example, type —dns and you'll see only DNS packets (Figure 7). When you start typing, Wireshark will help to you auto complete your filter.

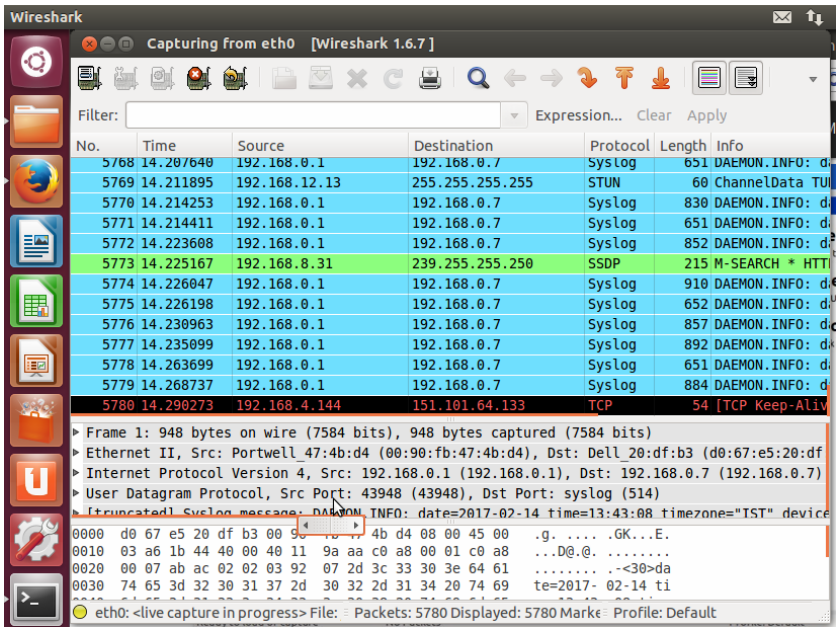


Figure 6: Filter in Wireshark and traffic running in Wireshark

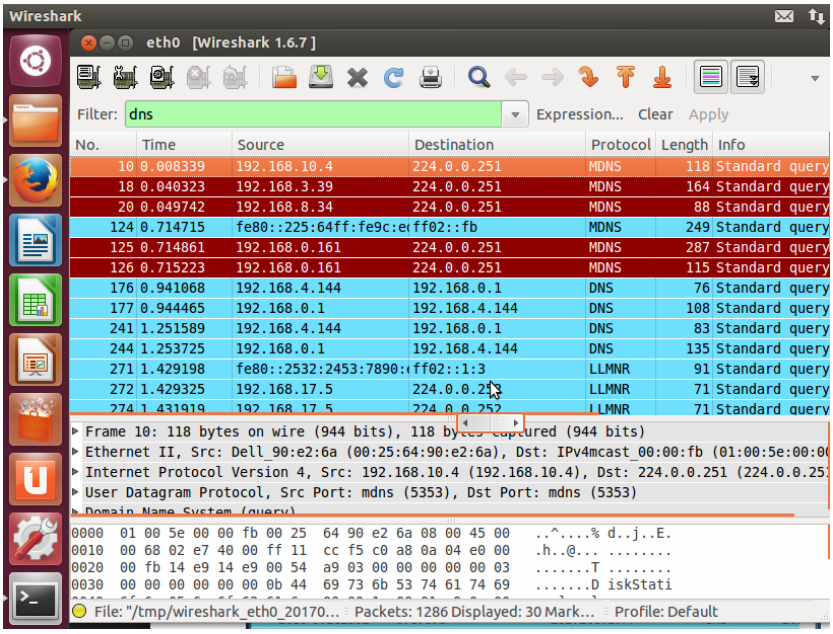


Figure 7: DNS Filter in Wireshark and traffic running in Wireshark

Program: Installation of Wireshark and running of Wireshark.

Output: Observe and analyze the traffic using Wireshark.

Conclusion: Hence we have successfully studied packet sniffer tool Wireshark.

Experiment No: 10

Aim: Download and install nmap. Use it with different options to scan open ports, perform OS fingerprinting, do a ping scan, tcp port scan, udp port scan, etc.

Theory:

Nmap (Network Mapper) is a security scanner originally written by Gordon Lyon (also known by his pseudonym Fyodor Vaskovich) used to discover hosts and services on a computer network, thus creating a "map" of the network. To accomplish its goal, Nmap sends specially crafted packets to the target host and then analyzes the responses. Unlike many simple port scanners that just send packets at some predefined constant rate, Nmap accounts for the network conditions (latency fluctuations, network congestion, the target interference with the scan) during the run. Also, owing to the large and active user community providing feedback and contributing to its features, Nmap has been able to extend its discovery capabilities beyond simply figuring out whether a host is up or down and which ports are open and closed; it can determine the operating system of the target, names and versions of the listening services, estimated uptime, type of device, and presence of a firewall.

Nmap features include:

- Host Discovery – Identifying hosts on a network. For example, listing the hosts which respond to pings or have a particular port open.
- Port Scanning – Enumerating the open ports on one or more target hosts.
- Version Detection – Interrogating listening network services listening on remote devices to determine the application name and version number.
- OS Detection – Remotely determining the operating system and some hardware characteristics of network devices.

Basic commands working in Nmap:

- For target specifications: nmap <target‘s URL or IP with spaces between them>
- For OS detection: nmap -O <target-host's URL or IP>
- For version detection: nmap -sV <target-host's URL or IP>

SYN scan is the default and most popular scan option for good reasons. It can be performed quickly, scanning thousands of ports per second on a fast network not hampered by restrictive firewalls. It is also relatively unobtrusive and stealthy since it never completes TCP connections

Algorithm\Implementation Steps\Installation Steps:

- Installing Nmap from the link.

  sudo apt-get install nmap

- Obtaining Your IP addresses.

  Use the ifconfig command in Linux.

- Performing a Scan of the Local Network.

1. For the following steps, please use the nmap command line tool installed on Ubuntu
2. Scan your subnet to determine how many hosts can be found. For example, if you are on the 192.168.1.0 subnet, you would enter the following command: nmap –sP 192.168.1.*

   i. What is your subnet? _____

ii. How many hosts were found? _____

3. Next perform a stealth scan (Please use the IP for your subnet): nmap –sS –P0 –p 192.169.1.*
4. Now, you'll perform an OS identification. Use the Linux O/S to scan your Windows machine:

   i. nmap –O Windows_IP_ADDRESS

ii. OS Type 1:_____

iii. Now we want to use the Windows machine to scan the Linux O/S. Go to a Windows DOS prompt and enter the following command:

iv. nmap –O Linux_IP_ADDRESS

v. Now we will perform a service selection scan. Let's scan for all computers with FTP running. We would do that as follows:

nmap –p21 192.168.1.*

5. List the IP addresses with that has the FTP open: _____

Program: Execution of nmap.

Output: Observe the output of namp.

Conclusion: Hence we have successfully studied and used nmap.

Experiment No: 11

Aim: Study of malicious software using different tools e.g. make use of the NESSUS to scan the network for vulnerabilities.

Theory:

Nessus is one of the most popular and capable vulnerability scanners, particularly for UNIX systems. Nessus is a remote security scanning tool, which scans a computer and raises an alert if it discovers any vulnerabilities that malicious hackers could use to gain access to any computer we have connected to a network.

Nessus is not a complete security solution, rather it is one small part of a good security strategy. NESSUS does not actively prevent attacks; it is only a tool that checks your computers to find vulnerabilities that hackers could exploit. It is up to the System Administrator to patch these Vulnerabilities in order to create a security solution.

To learn how NESSUS and other port-scanning security tools work, it is necessary to understand different services (such as a web server, SMTP server, FTP server, etc) are accessed on a remote server. Most high-level network traffic, such as email, web pages, etc reach a server via a high-level protocol that is transmitted reliably by a TCP stream. To keep different streams from interfering with each other, a computer divides its physical connection to the network into thousands of logical paths, called ports. So if we want to talk to a web server on a given machine, we would connect to port #80 (the standard HTTP port), but if we want to connect to an SMTP server on that same machine we would instead connect to port #25.

Each computer has thousands of ports, all of which may or may not have services (ie: a server for a specific high-level protocol) listening on them. Nessus works by testing each port on a computer, determining what service it is running, and then testing this service to make sure there are no vulnerabilities in it that could be used by a hacker to carry out a malicious attack. Nessus is called a "remote scanner" because it does not need to be installed on a computer for it to test that computer. Instead, we can install it on only one computer and test as many computers as we would like.

Conclusion: Hence we have successfully studied a security vulnerability scanning tool Nessus.

Experiment No: 12

Aim: Study of Network security by setting up IPSEC under LINUX

Theory:

Internet Protocol Security (**IPsec**) is a protocol suite for securing Internet Protocol (IP) communications by authenticating and encrypting each IP packet of a communication session. IPsec includes protocols for establishing mutual authentication between agents at the beginning of the session and negotiation of cryptographic keys to be used during the session. IPsec can be used in protecting data flows between a pair of hosts (*host-to-host*), between a pair of security gateways (*network-to-network*), or between a security gateway and a host (*network-to-host*)
Internet Protocol security (IPsec) uses cryptographic security services to protect communications over Internet Protocol (IP) networks. IPsec supports network-level peer authentication, data origin authentication, data integrity, and data confidentiality (encryption), and replay protection.
IPsec is an end-to-end security scheme operating in the Internet Layer of the Internet Protocol Suite, while some other Internet security systems in widespread use, such as Transport Layer Security (TLS) and Secure Shell (SSH), operate in the upper layers at Application layer. Hence, only IPsec protects any application traffic over an IP network. Applications can be automatically secured by IPsec at the IP layer.

ipsec can be implemented using strongSwan tool. strongSwan is a IPsec implementation. It uses openSSL pluugin (Eliptic Curve Cryptography).

Let two servers be red server (192.168.4.144) and blue server (192.168.4.145)

Installation and configuration on red server

Step 1: Installation of strongswan

project@project-OptiPlex-360:~$ sudo apt-get install ipsec-tools strongswan-starter

Step 2: Configuration of ipsec.conf

project@project-OptiPlex-360:~$ sudo gedit /etc/ipsec.conf

Step 3: Configuration of ipsec.secrets

project@project-OptiPlex-360:~$ sudo gedit /etc/ipsec.secrets

Step 4: Start ipsec

project@project-OptiPlex-360:~$ sudo ipsec restart

Stopping strongSwan IPsec...

Starting strongSwan 5.3.5 IPsec [starter]...

Step 5: Checking status information of ipsec

project@project-OptiPlex-360:~$ sudo ipsec statusall

Conclusion: Hence we have successfully studied IPSec.

Name : Kamal Agrahari **Class:**TE/IT/A

ID NO: VU4F2223028 **Lab:** PCE II

Assignment.No.06.

Ques) Prepare documentation for conducting meetings on any of the following topics? (prepare notice, agenda and minutes of meeting) -

1. The annual meeting of the housing society.
2. Meeting regarding Annual Function in your college.
3. The meeting of the Board of Directors in organization.

Ans:

VASANTDADA PATIL PRATISHTHAN'S COLLEGE OF ENGINEERING AND VISUAL ARTS

Department of Information Technology

Date:23/09/2024

Notice

Notice to inform you that a meeting has been scheduled to discuss and plan the upcoming **Annual Function** of the college. The details of the meeting are as follows:

Date: 26/09/2024
Time:05:00 PM to 09:00 PM
Venue: VPPCOE & VA Campus

The agenda for the meeting is enclosed below. You are requested to make yourself available for the meeting as your participation and input are essential for the successful organization of the event.

Sincerely,
Kamal Agrahari
Head of Cultural Committee
7021xxxx60

To: 1) Principal Office,
2) HODs - IT/CS/AiDs,
3) Faculty Members,
4) Student Council

Enclosed: Agenda of meeting

VASANTDADA PATIL PRATISHTHAN'S COLLEGE OF ENGINEERING AND VISUAL ARTS
Department of Information Technology

Agenda for the Meeting Regarding the Annual Function

Date:26/09/2024
Time:05:00 PM to 09:00 PM
Venue: Seminar Hall

Meeting Agenda:

1. **Welcome Address by the Chairperson**
    - Overview of the previous annual functions
    - Purpose of the meeting
2. **Review of Last Year's Annual Function**
    - Successes
    - Areas for improvement
3. **Discussion on the Theme of the Event**
    - Proposal of themes
    - Selection of the final theme
4. **Allocation of Responsibilities**
    - Cultural programs (dance, music, drama, etc.)
    - Invitations (Chief Guest, Guests of Honor, etc.)
    - Decoration and Stage Setup
    - Budget and Sponsorship Committee
    - Security and Logistics
5. **Budget Discussion**
    - Estimation of funds required
    - Sponsorship opportunities
6. **Selection of Chief Guest and Special Guests**
    - Suggestions for prominent figures to be invited
7. **Cultural Program Schedule**
    - Setting the sequence of events
8. **Event Promotion**
    - Marketing strategies
    - Social media and print publicity
9. **Feedback and Suggestions**
    - Open forum for suggestions from faculty and student representatives
10. **Conclusion and Closing Remarks**

Thank you,
Kamal Agrahari
Head of Cultural Committee
7021xxxx60

VASANTDADA PATIL PRATISHTHAN'S COLLEGE OF ENGINEERING AND VISUAL ARTS

Department of Information Technology

Regarding Board of Directors Meeting on Annual Function

Minutes

On the board of directors for the annual function held on **Monday, 23 September, 2024 at 10:00 AM**. at the Ground Floor, Vasantdada Patil Education Complex, Near Eastern Express Highway, Sion Chunabhatti, Mumbai, India - 400028

Present:

- Mr. Rutvik Gondekar (Chairperson)
- Mr. Ritesh Maurya (Board Member)
- Mr. Akash Nahak (Board Member)
- Mr. Yogiraj Shinde (Board Member)
- Mr. Vijay Sharma (Board Member)
- Mr. Kamal Agrahari (Secretary)

No of Minutes	Subject of Minutes	Details of minute
1	Welcome Address by the Chairperson	The Chairperson welcomed all attendees and provided an overview of the previous year's annual function. The purpose of the meeting was outlined.
2	Review of Last Year's Annual Function	Successes and challenges from last year's event were discussed. Focus on time management and event flow was agreed upon for improvement.
3	Discussion on the Theme of the Event	Three themes were proposed. The theme "Celebrating Diversity" was unanimously chosen after discussion.
4	Allocation of Responsibilities	Responsibilities were allocated for Cultural Programs, Invitations, Decoration, Budget, and Security to respective team members.
5	Budget Discussion	An initial budget was proposed. External sponsorships will be sought from local businesses by the sponsorship team.
6	Selection of Chief Guest and Guests	Suggestions for the Chief Guest were made, with final decision pending confirmation from the administration.
7	Cultural Program Schedule	A draft schedule was created. The final schedule will be confirmed after all performances are finalized.
8	Event Promotion	A promotional campaign involving posters, social media, and local media outreach was decided.
9	Feedback and Suggestions	Attendees suggested adding a talent show and increasing student participation.
10	Conclusion and Closing Remarks	The Chairperson thanked everyone,adjourned the meeting, and scheduled the next one for 25/09/2024.

Dated:23/09/2024

Mr. Rutvik Gondekar
Chairperson

Mr. Kamal Agrahari
Secretary