

Name: Love Sharma  
Sap-Id: 500060174  
Roll: 32  
Subject: System Provisioning And Configuration System  
Mid-Sem Project  
Course: B-tech CSE Devops

## 1. Create a small web application including DataBase

This application is a Register and Login functionality.

Welcome to my Flask Application	Home	About	Register	Login
---------------------------------	------	-------	----------	-------

# System Provisioning Midsem Project

Basic Python Flask Authentication Page

[Register](#)[Login](#)

Welcome to my Flask Application	Home	About	Register	Login
---------------------------------	------	-------	----------	-------

## Register

Name

Email

Username

Password

Confirm password

You are now logged in >

# Dashboard

Welcome love

## YOU ARE CURRENTLY LOGGED IN

### 2. Create a job in Jenkins to make build of this application.

Workflow:

#### Two jobs have been created

**Job 1:** This job deals with fetching image from git repository, then building the image and at last deploying the built image to dockerhub repository.

**Job 2:** This job is triggered if the job 1 is successful. This job deploys the docker instances on the local machine using docker-compose.

#### JOB 1 (Image-Build)

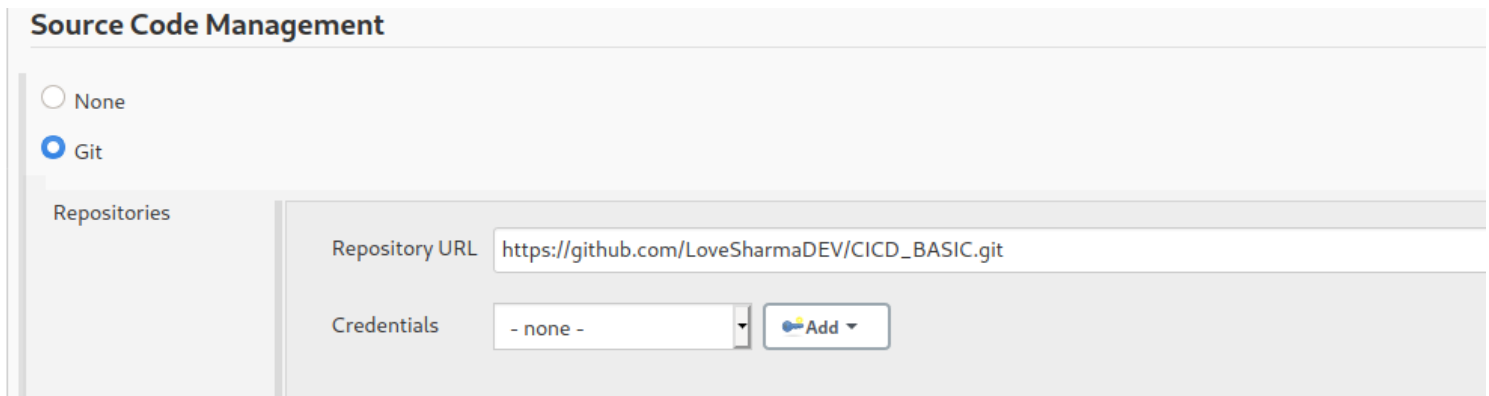
Docker plugin Used:



#### CloudBees Docker Build and Publish plugin

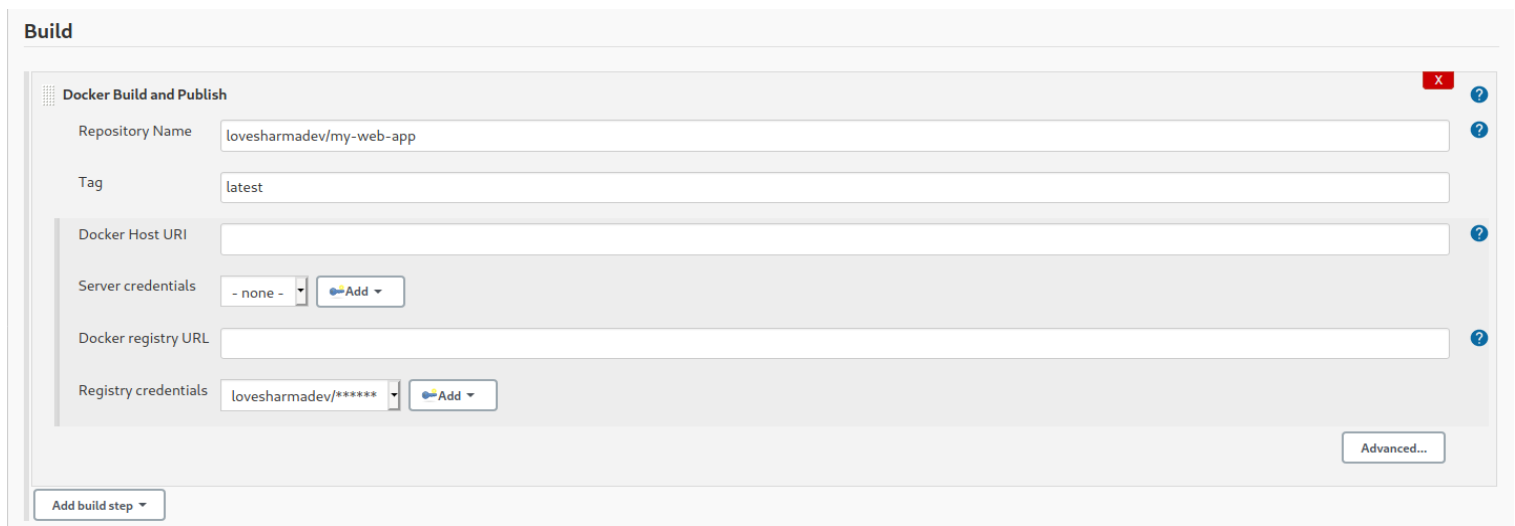
This plugin enables building Dockerfile based projects, as well as publishing of the built images/repos to the docker registry.

1. provide **Jenkins** the repository to fetch code from



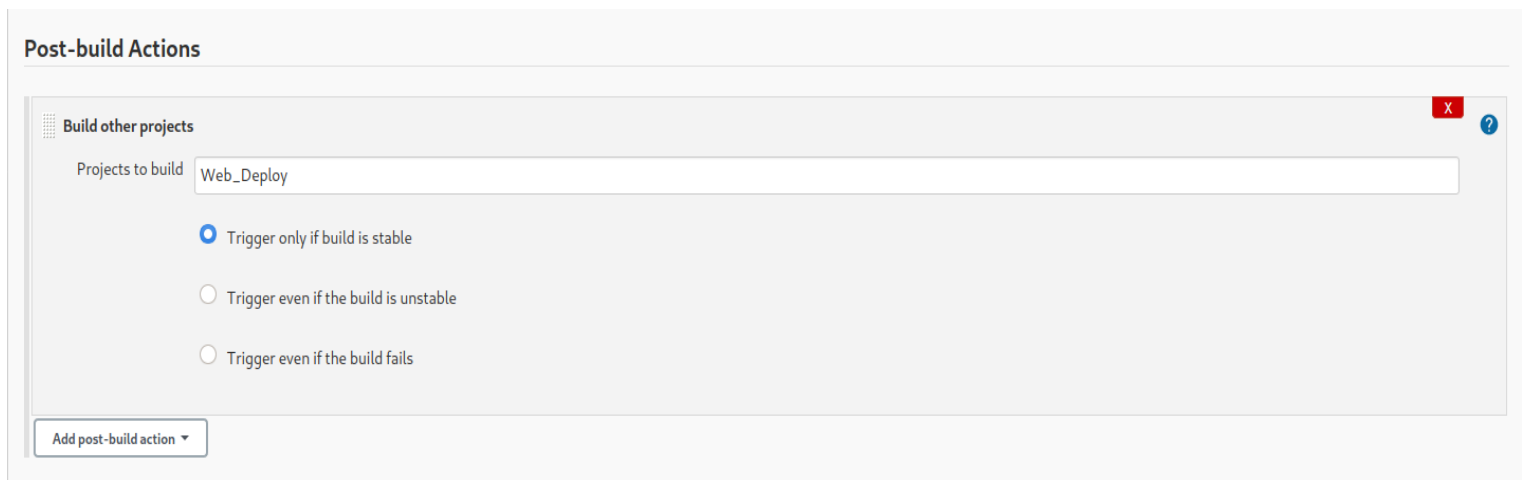
The screenshot shows the 'Source Code Management' configuration page in Jenkins. The 'Git' option is selected under the 'None' and 'Git' radio buttons. In the 'Repositories' section, the 'Repository URL' is set to 'https://github.com/LoveSharmaDEV/CICD\_BASIC.git'. The 'Credentials' dropdown is set to '- none -' with an 'Add' button next to it.

2. Use the **plugin** to configure Jenkins to dockerhub repository



The screenshot shows the 'Build' configuration page in Jenkins. The 'Docker Build and Publish' section is expanded. The 'Repository Name' is 'lovesharmadev/my-web-app' and the 'Tag' is 'latest'. The 'Docker Host URI' is empty. The 'Server credentials' dropdown is set to '- none -' with an 'Add' button. The 'Docker registry URL' is empty. The 'Registry credentials' dropdown is set to 'lovesharmadev/\*\*\*\*\*' with an 'Add' button. There is an 'Advanced...' button at the bottom right of the section and an 'Add build step' button at the bottom left.

3. Set the post build actions to **trigger** the deployment build




The screenshot shows the 'Post-build Actions' configuration page in Jenkins. The 'Build other projects' section is expanded. The 'Projects to build' dropdown is set to 'Web\_Deploy'. There are three radio button options: 'Trigger only if build is stable' (selected), 'Trigger even if the build is unstable', and 'Trigger even if the build fails'. There is an 'Add post-build action' button at the bottom left.

## JOB 2 (Image-Deploy to local environment) ==> STAGING ENVIRONMENT)

Deploy using **docker compose**

**Build**

 **Execute shell**

Command

```
cd /var/lib/jenkins/workspace/Docker-Build-Job/  
docker-compose down -v --rmi local  
docker-compose up -d
```

[See the list of available environment variables](#)

## SCREEN SHOTS

### Console Output

```
Started by user Love Sharma
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/Docker-Build-Job
The recommended git tool is: NONE
No credentials specified
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/LoveSharmaDEV/CICD_BASIC.git # timeout=10
Fetching upstream changes from https://github.com/LoveSharmaDEV/CICD_BASIC.git
> git --version # timeout=10
> git --version # 'git version 2.28.0'
> git fetch --tags --force --progress -- https://github.com/LoveSharmaDEV/CICD_BASIC.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision edb3fd0bd28b4c8f74cf8759281af15a3ab7987f (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f edb3fd0bd28b4c8f74cf8759281af15a3ab7987f # timeout=10
Commit message: "Updates"
First time build. Skipping changelog.
[Docker-Build-Job] $ docker build -t lovesharmadev/my-web-app --pull=true /var/lib/jenkins/workspace/Docker-Build-Job
Sending build context to Docker daemon 2.169MB

Step 1/8 : FROM python
latest: Pulling from library/python
Digest: sha256:03af1810bd46d201a4bd1477aeff337a09e67118d96c50138e3360c2b4bd6b9f
Status: Image is up to date for python:latest
--> dfc47c6cee13
Step 2/8 : RUN apt-get update -y
--> Using cache
--> 0fde1e2ff1f0
Step 3/8 : RUN pip install Flask
--> Using cache
--> 256d116640a1
```

### Console Output

```
Started by upstream project "Docker-Build-Job" build number 10
originally caused by:
  Started by user Love Sharma
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/Web_Deploy
[Web_Deploy] $ /bin/sh -xe /tmp/jenkins8912279571567638141.sh
+ cd /var/lib/jenkins/workspace/Docker-Build-Job/
+ docker-compose down -v --rmi local
Removing network docker-build-job_default
+ docker-compose up -d
Creating network "docker-build-job_default" with the default driver
Creating docker-build-job_database_1 ...
[1A]2K
Creating docker-build-job_database_1 ... [32mdone[0m
[1BCreating docker-build-job_webapp_1 ...
[1A]2K
Creating docker-build-job_webapp_1 ... [32mdone[0m
[1BFinished: SUCCESS
```

All

+

S	W	Name ↓	Last Success	Last Failure	Last Duration	
		Docker-Build-Job	1 min 13 sec - #1 lovesharmadev/my-web-app	N/A	29 sec	
		Web_Deploy	37 sec - #1	N/A	20 sec	

Icon: S M L

Legend
Atom feed for all
Atom feed for failures
Atom feed for just latest builds

```
love@kali ~
$ docker ps
CONTAINER ID        IMAGE               NAMES               COMMAND               CREATED             STATUS
5a42b4b701d0        lovesharmadev/my-php-app   "/docker-entrypoin...   About a minute ago   Up About
a minute          0.0.0.0:8000->80/tcp   docker-build-job_webapp_1
5d11d8d5f1af        mysql:5.7            "docker-entrypoin.s...   2 minutes ago        Up About
a minute          3306/tcp, 33060/tcp   docker-build-job_database_1
love@kali ~
$
```

3. Write a Terraform script to deploy the image on AWS Docker platform.

#### BRIEF ON STEPS:

1. Initialize the provider to set the API's to be used. After setting "aws" as a provider you would be able to use the AWS resources.

```
provider "aws"{
  region="us-east-1"
  access_key="*****"
  secret_key="*****"
}
```

2. Next step is to set up the VPC. This is the place where we would be deploying our ECS cluster

```
resource "aws_vpc" "main"{
  cidr_block = "132.0.0.0/16"
  tags = {
    Name=var.vpc_name
  }
}
```

3. After creating VPC we would have to setup the subnet configuration.

```
resource "aws_subnet" "main" {
  count = 2
  vpc_id      = aws_vpc.main.id
  cidr_block  = cidrsubnet(aws_vpc.main.cidr_block, 8, count.index)
  map_public_ip_on_launch=true
  tags = {
    Name = var.subnet_name
  }
}
```

4. Next step involves configuration of our vpc and subnets.

```
resource "aws_subnet" "main" {
  count = 2
  vpc_id      = aws_vpc.main.id
  cidr_block = cidrsubnet(aws_vpc.main.cidr_block, 8, count.index)
  map_public_ip_on_launch=true
  tags = {
    Name = var.subnet_name
  }
}

resource "aws_internet_gateway" "internetgateway" {
  vpc_id = aws_vpc.main.id
}

resource "aws_route" "internet_access" {
  route_table_id = aws_vpc.main.main_route_table_id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id = aws_internet_gateway.internetgateway.id
}
```

5. The next step is to implement AWS security groups for incoming traffic and to configure permissions ECS by IAM policies.



```

resource "aws_security_group" "accessgroups" {
  name = "allowinbound"
  vpc_id = aws_vpc.main.id
  ingress {
    cidr_blocks=["0.0.0.0/0"]
    from_port=0
    to_port=65535
    protocol="tcp"
  }
  ingress {
    cidr_blocks=["0.0.0.0/0"]
    from_port=0
    to_port=0
    protocol=-1
  }
  tags = {
    Name = "ECS-Access"
  }
}

data "aws_iam_policy_document" "ecs_task_execution_role" {
  version = "2012-10-17"
  statement {
    sid = ""
    effect = "Allow"
    actions = ["sts:AssumeRole"]

    principals {
      type       = "Service"
      identifiers = ["ecs-tasks.amazonaws.com"]
    }
  }
}

```

**6. Last step boils all down to configuring ECS**

```

# ECS task execution role
resource "aws_iam_role" "ecs_task_execution_role" {
  name           = "MyEcsTaskExecutionRole"
  assume_role_policy = data.aws_iam_policy_document.ecs_task_execution_role.json
}

# ECS task execution role policy attachment
resource "aws_iam_role_policy_attachment" "ecs_task_execution_role" {
  role       = aws_iam_role.ecs_task_execution_role.name
  policy_arn = "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy"
}

resource "aws_ecs_cluster" "nodecluster" {
  name = "sysprov"
}

resource "aws_ecs_task_definition" "flaskapp" {
  family           = "service"
  container_definitions = file("service.json")
  execution_role_arn=aws_iam_role.ecs_task_execution_role.arn
  network_mode="awsvpc"
  requires_compatibilities=["FARGATE"]
  memory="1024"
  cpu="512"
}

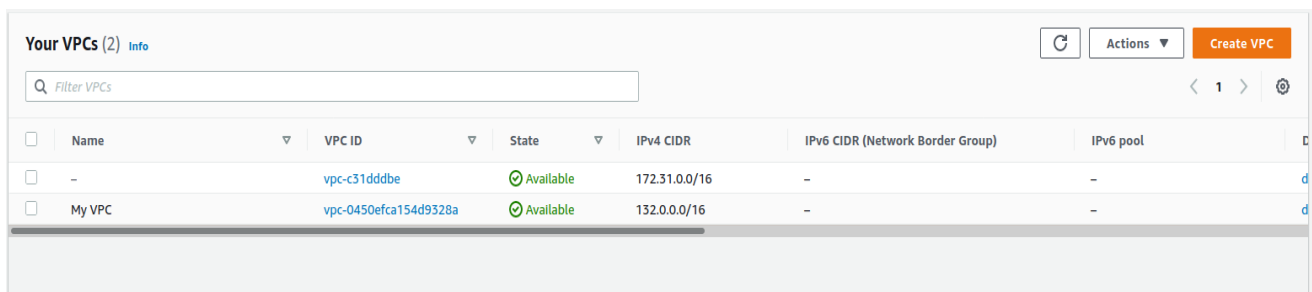
resource "aws_ecs_service" "main" {
  name = "service-ecs"
  cluster = aws_ecs_cluster.nodecluster.name
  task_definition = aws_ecs_task_definition.flaskapp.arn
  launch_type = "FARGATE"
  network_configuration {
    security_groups = [aws_security_group.accessgroups.id]
    subnets = aws_subnet.main.*.id
    assign_public_ip = true
  }
  depends_on=[aws_iam_role_policy_attachment.ecs_task_execution_role]
}

```

**AFTER CREATING TERRAFORM CONFIG FILES  
RUN FOLLOWING COMMANDS ON THE TERMINAL:**

1. terraform init
2. terraform plan
3. terraform apply

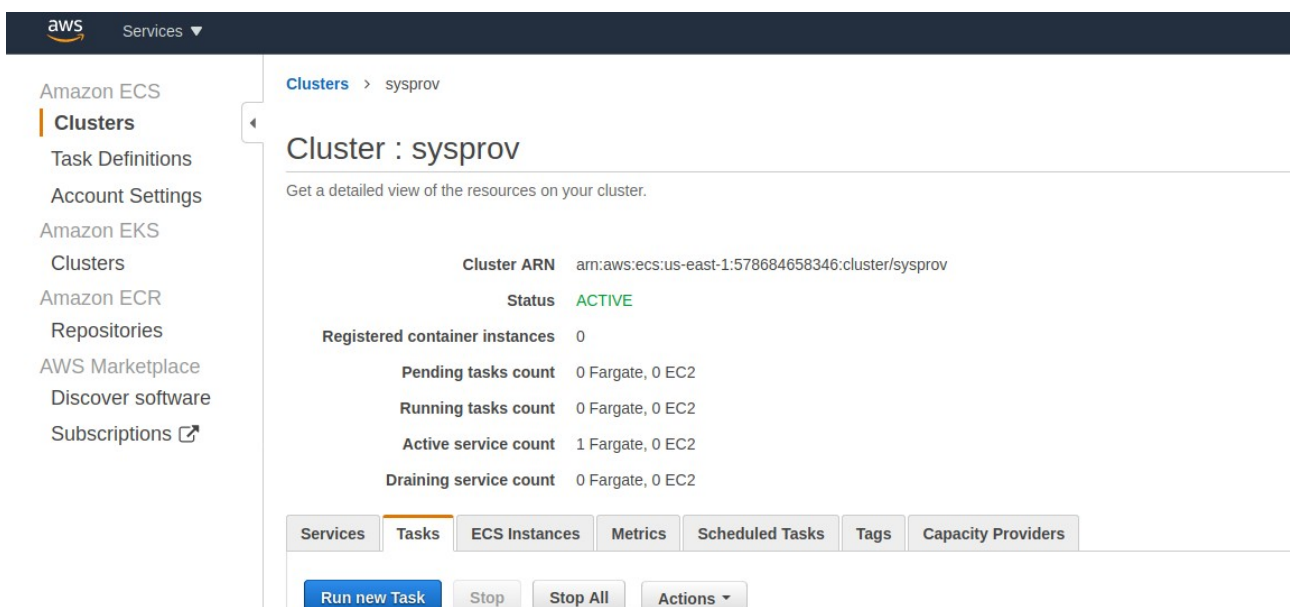
On running above commands the architecture defined in terraform would be provisioned on aws platform.



The screenshot shows the AWS VPC console interface. At the top, it says "Your VPCs (2)" with an "Info" link. There is a search bar labeled "Filter VPCs" and buttons for "Actions" and "Create VPC". Below this is a table listing the VPCs.

<input type="checkbox"/>	Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR (Network Border Group)	IPv6 pool
<input type="checkbox"/>	-	vpc-c31dddbb	Available	172.31.0.0/16	-	-
<input type="checkbox"/>	My VPC	vpc-0450efca154d9328a	Available	132.0.0.0/16	-	-

Next task would be to create a task and run it on ECS cluster.



The screenshot shows the AWS ECS console. The left sidebar lists navigation options: Amazon ECS, Clusters, Task Definitions, Account Settings, Amazon EKS, Clusters, Amazon ECR, Repositories, AWS Marketplace, Discover software, and Subscriptions. The main content area shows the "Clusters" page with a breadcrumb "Clusters > sysprov". The title is "Cluster : sysprov". Below the title, it says "Get a detailed view of the resources on your cluster." A table displays cluster details:

Cluster ARN	arn:aws:ecs:us-east-1:578684658346:cluster/sysprov
Status	ACTIVE
Registered container instances	0
Pending tasks count	0 Fargate, 0 EC2
Running tasks count	0 Fargate, 0 EC2
Active service count	1 Fargate, 0 EC2
Draining service count	0 Fargate, 0 EC2

At the bottom, there are tabs for "Services", "Tasks", "ECS Instances", "Metrics", "Scheduled Tasks", "Tags", and "Capacity Providers". Below the tabs are buttons: "Run new Task", "Stop", "Stop All", and "Actions".

## Run Task

Select the cluster to run your task definition on and the number of copies of that task to run. To apply container overrides or target particular container instances, click **Advanced Options**.

**Launch type** ☒ FARGATE ☐ EC2 


[Switch to capacity provider strategy](#) 

**Task Definition** Family:    
Revision:  

**Platform version**  


**Cluster**


**Number of tasks**

**Task Group**  

### VPC and security groups

VPC and security groups are configurable when your task definition uses the awsvpc network mode.

**Cluster VPC\***  

**Subnets\***  

**Security groups\***   

**Auto-assign public IP**  

[Advanced Options](#)

Amazon ECS

Clusters

Task Definitions

Account Settings

Amazon EKS


Clusters

Amazon ECR

Repositories


AWS Marketplace


Discover software



Subscriptions 


## Run Task

Select the cluster to run your task definition on and the number of copies of that task to run. To apply container overrides or target particular container instances, click **Advanced Options**.

**Launch type** ☒ FARGATE ☐ EC2 


[Switch to capacity provider strategy](#) 

**Task Definition** Family:    
Revision:  

**Platform version**  


**Cluster**


**Number of tasks**


**Task Group**  


### VPC and security groups

VPC and security groups are configurable when your task definition uses the awsvpc network mode.


**Cluster VPC\***  

**Subnets\***  

**Security groups\***   

**Auto-assign public IP**  


[Advanced Options](#)


 **Tagging requires that you opt in to the new ARN and resource ID format.**  
The old userguide has not opted in to the new ARN format. Opt in to the new format to use this feature. [Learn more.](#)

### Configure security groups

A security group is a set of firewall rules that control the traffic for your task. On this page, you can add rules to allow specific traffic to reach your task, or you can choose to use an existing security group. [Learn more.](#)

**Assigned security groups** ☒ Create new security group  
☐ Select existing security group

**Security group name\***  

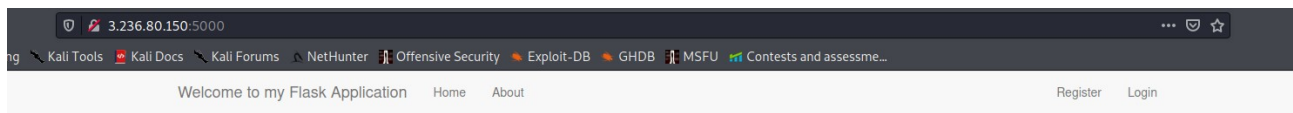
**Description**  

#### Inbound rules for security group

Type	Protocol	Port range	Source	
HTTP	TCP	80	Anywhere	0.0.0.0/0 ::0
All TCP	TCP	0 - 65535	Anywhere	0.0.0.0/0 ::0
All traffic	All	0 - 65535	Anywhere	0.0.0.0/0 ::0

[Add rule](#)

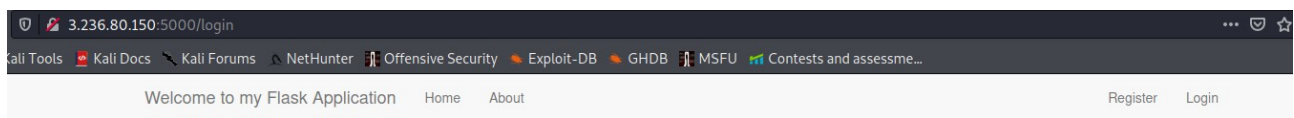
After all of the configuration, access your task through public IP provided.



# System Provisioning Midsem Project

Basic Python Flask Authentication Page

[Register](#) [Login](#)



## Login

**Username**

**Password**

[Submit](#)