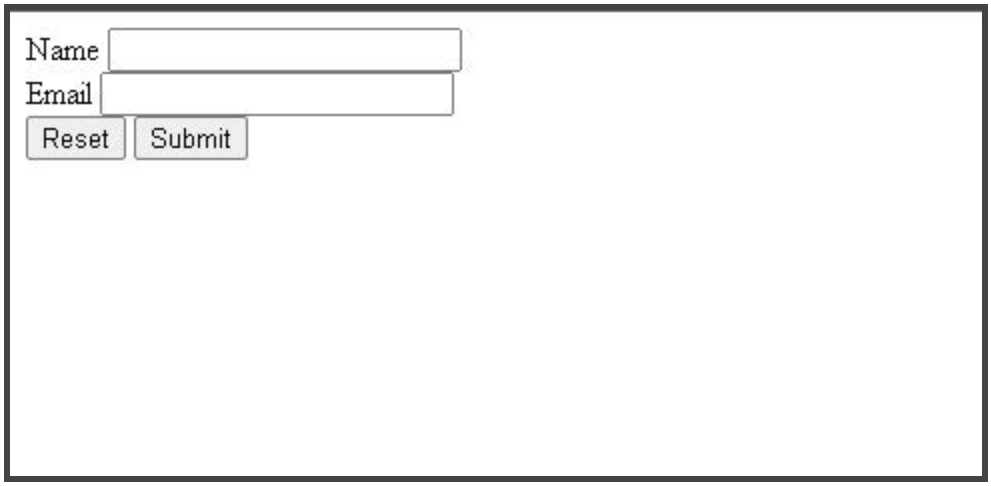
Midsem Project
SYSTEM PROVISIONING AND CONFIGURATION SYSTEM

Name: Mudit Garg
Roll No: R171217036
Sap ID: 500062539

1. Create a small web application




2. Create a Dockerfile
   A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image

```
vagrant@vagrant-ubuntu-trusty-64:~/DockerFiles$ cat Dockerfile
FROM phpmyadmin:latest

COPY form.html /var/www/html/form.html
COPY apache2.conf /etc/apache2/apache2.conf
#ENTRYPOINT bash
vagrant@vagrant-ubuntu-trusty-64:~/DockerFiles$
```

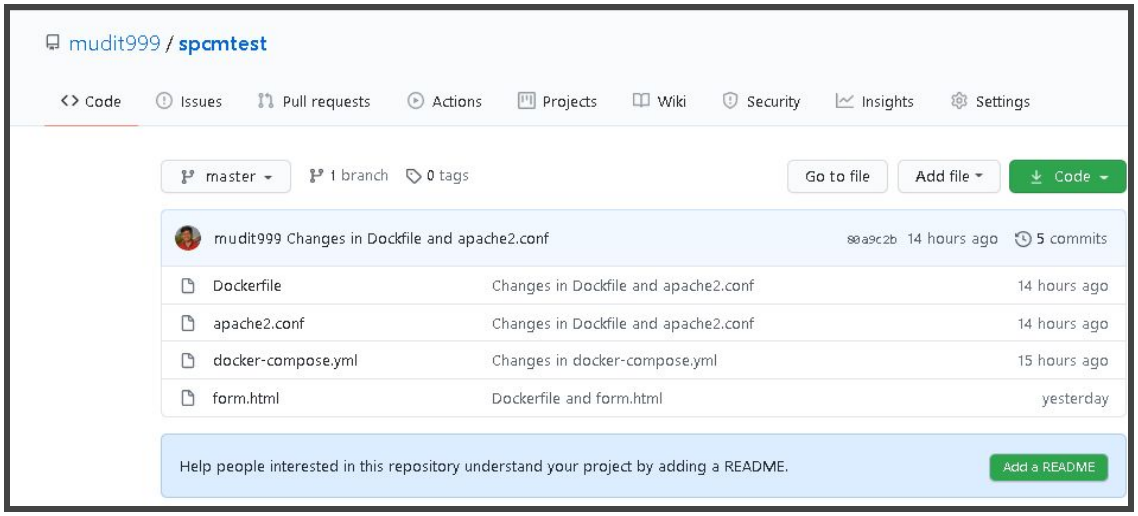
3. Create apache2.configuration file

```
vagrant@vagrant-ubuntu-trusty-64:~/DockerFiles$ cat apache2.conf
# This is the main Apache server configuration file.  It contains the
# configuration directives that give the server its instructions.
# See http://httpd.apache.org/docs/2.4/ for detailed information about
# the directives and /usr/share/doc/apache2/README.Debian about Debian specific
# hints.
#
#
# Summary of how the Apache 2 configuration works in Debian:
# The Apache 2 web server configuration in Debian is quite different to
# upstream's suggested way to configure the web server. This is because Debian's
# default Apache2 installation attempts to make adding and removing modules,
# virtual hosts, and extra configuration directives as flexible as possible, in
# order to make automating the changes and administering the server as easy as
# possible.

# It is split into several files forming the configuration hierarchy outlined
# below, all located in the /etc/apache2/ directory:
#
#       /etc/apache2/
#       |-- apache2.conf
#       |       `--  ports.conf
#       |-- mods-enabled
#       |       |-- *.load
#       |       `-- *.conf
#       |-- conf-enabled
#       |       `-- *.conf
#       `-- sites-enabled
#               `-- *.conf
```

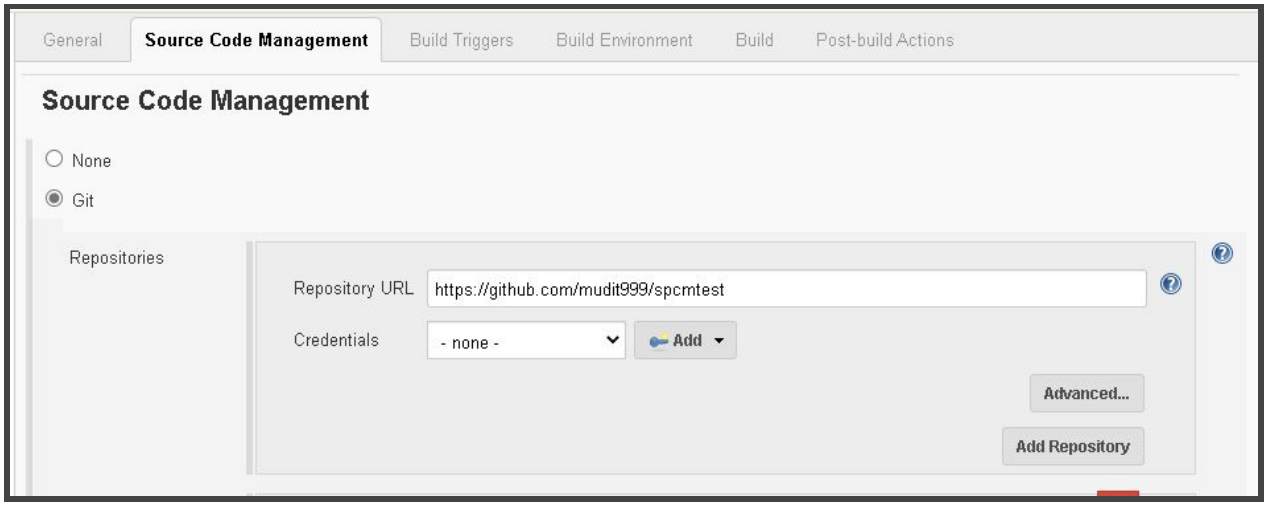
4. Create Docker Compose file

```
vagrant@vagrant-ubuntu-trusty-64:~/DockerFiles$ cat docker-compose.yml
version: '3.3'

services:
   webapp:
     image: mudit999/spcm-test-repo
     ports:
       - "8000:80"
     restart: always
vagrant@vagrant-ubuntu-trusty-64:~/DockerFiles$
```

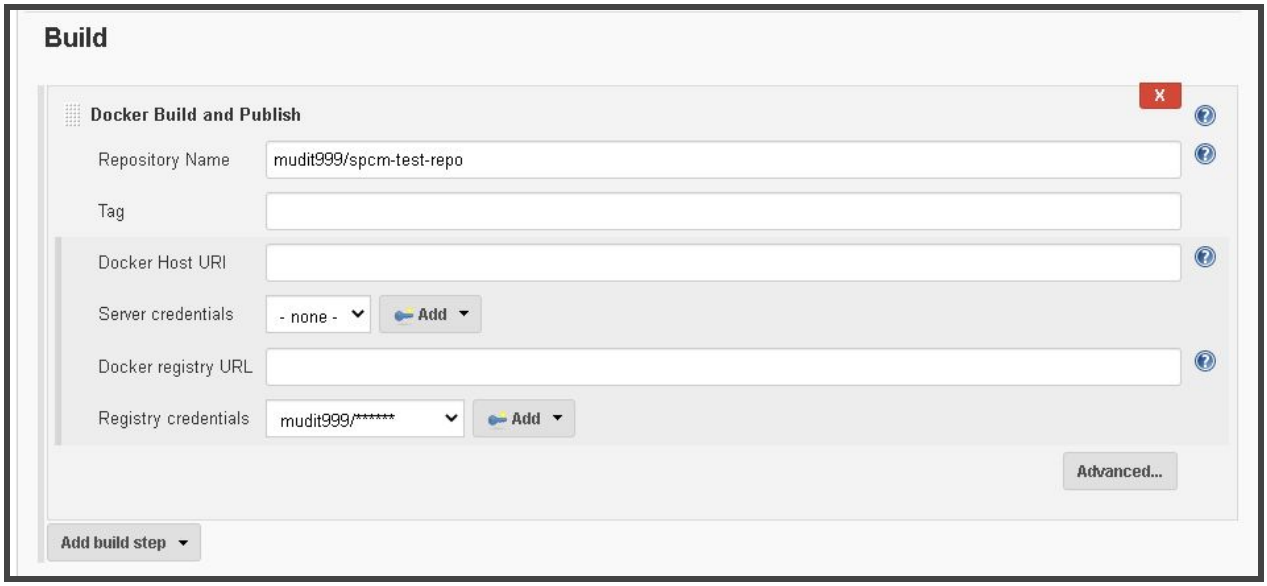
5. Push these files to github




6. Create a job on jenkins
   - Provide github repo Url



- Install Docker plugin: Docker Build and Publish



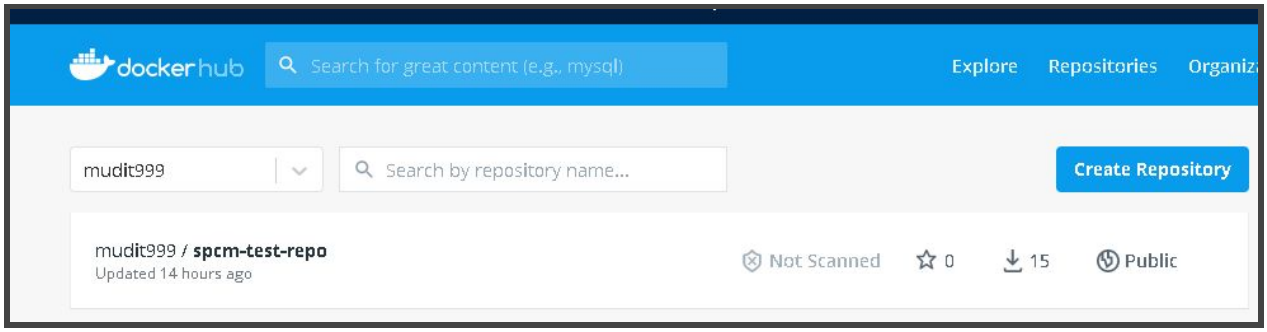

- Then build this job on Jenkins

# Console Output

```
Started by user Mudit Garg
Running as SYSTEM
Building in workspace /root/.jenkins/workspace/spcm-test
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/mudit999/spcmtest.git # timeout=10
Fetching upstream changes from https://github.com/mudit999/spcmtest.git
 > git --version # timeout=10
 > git --version # 'git version 2.24.0'
 > git fetch --tags --force --progress -- https://github.com/mudit999/spcmtest.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 80a9c2bf4f98299ddc202878f11d03fbd0a1bc9e (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 80a9c2bf4f98299ddc202878f11d03fbd0a1bc9e # timeout=10
Commit message: "Changes in Dockfile and apache2.conf"
 > git rev-list --no-walk 80a9c2bf4f98299ddc202878f11d03fbd0a1bc9e # timeout=10
[spcm-test] $ docker build -t mudit999/spcm-test-repo --pull=true /root/.jenkins/workspace/spcm-test
Sending build context to Docker daemon   93.7kB

Step 1/3 : FROM phpmyadmin:latest
latest: Pulling from library/phpmyadmin
Digest: sha256:5ae73d0a421e4bb7284640e0a7792709068f54866d7dce4d8396f58fa9e76deb
Status: Image is up to date for phpmyadmin:latest
 ---> 5fa0b0c8888e
Step 2/3 : COPY form.html /var/www/html/form.html
 ---> Using cache
 ---> c968d4ffd3d0
Step 3/3 : COPY apache2.conf /etc/apache2/apache2.conf
 ---> Using cache
 ---> ac6e55c33687
Successfully built ac6e55c33687
Successfully tagged mudit999/spcm-test-repo:latest
[spcm-test] $ docker inspect ac6e55c33687
[spcm-test] $ docker push mudit999/spcm-test-repo
The push refers to repository [docker.io/mudit999/spcm-test-repo]
433b47d0495e: Preparing
705b999c878c: Preparing
9542cac5d52a: Preparing
71d45b84e621: Preparing
6da1c072b05a: Preparing
dd0576a7d248: Preparing
ae8cd72e3c62: Preparing
13d54f76b02e: Preparing
2f2e73839893: Preparing
82801ee01d09: Preparing
```

```
2f2e73839893: Preparing
82801ee01d09: Preparing
f447060f9935: Preparing
7d55cc606fe1: Preparing
c9ec7fe9a88a: Preparing
435f9e9c9e58: Preparing
fc0b35927b5a: Preparing
8c7a39edfbf0: Preparing
8888f89886ba: Preparing
f182865e86e9: Preparing
b2bb5e569df9: Preparing
d0fe97fa8b8c: Preparing
dd0576a7d248: Waiting
13d54f76b02e: Waiting
2f2e73839893: Waiting
82801ee01d09: Waiting
f447060f9935: Waiting
7d55cc606fe1: Waiting
c9ec7fe9a88a: Waiting
435f9e9c9e58: Waiting
fc0b35927b5a: Waiting
8c7a39edfbf0: Waiting
8888f89886ba: Waiting
f182865e86e9: Waiting
b2bb5e569df9: Waiting
d0fe97fa8b8c: Waiting
ae8cd72e3c62: Waiting
71d45b84e621: Layer already exists
705b999c878c: Layer already exists
433b47d0495e: Layer already exists
9542cac5d52a: Layer already exists
6da1c072b05a: Layer already exists
dd0576a7d248: Layer already exists
13d54f76b02e: Layer already exists
ae8cd72e3c62: Layer already exists
82801ee01d09: Layer already exists
f447060f9935: Layer already exists
7d55cc606fe1: Layer already exists
435f9e9c9e58: Layer already exists
fc0b35927b5a: Layer already exists
2f2e73839893: Layer already exists
8c7a39edfbf0: Layer already exists
8888f89886ba: Layer already exists
c9ec7fe9a88a: Layer already exists
b2bb5e569df9: Layer already exists
d0fe97fa8b8c: Layer already exists
f182865e86e9: Layer already exists
latest: digest: sha256:d7c696be41dc43d26e7023ea49e475c15aa24c61e90d06ab4c6117959526f465 size: 4495
Finished: SUCCESS
```

Here the web application image will be created and pushed on Docker hub




7. Write a terraform script to deploy this image on AWS

Add access key and secret key of AWS account

```
vagrant@vagrant-ubuntu-trusty-64:~/DockerFiles/terraform$ cat main.tf
provider "aws"{
    region="us-east-1"
    access_key="[redacted]"
    secret_key="[redacted]"
}
```

Here add docker image

```
vagrant@vagrant-ubuntu-trusty-64:~/DockerFiles/terraform$ cat service.json
[
    {
      "name": "first",
      "image": "docker.io/mudit999/spcm-test-repo",
      "cpu": 10,
      "memory": 512,
      "essential": true,
      "portMappings": [
        {
          "containerPort": 5001,
          "hostPort": 5001
        }
      ]
    }
]
```

```
vagrant@vagrant-ubuntu-trusty-64:~/DockerFiles/terraform$ cat variable.tf
variable "vpc_name"{
    default="My VPC"
}
variable "subnet_name"{
    default="My Subnet"
}
```

Then apply these terraform commands:

1. terraform init

```
vagrant@vagrant-ubuntu-trusty-64:~/DockerFiles/terraform$ vim main.tf
vagrant@vagrant-ubuntu-trusty-64:~/DockerFiles/terraform$ terraform init

Initializing the backend...

Initializing provider plugins...
- Checking for available provider plugins...
- Downloading plugin for provider "aws" (hashicorp/aws) 3.11.0...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.aws: version = "~> 3.11"

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. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
vagrant@vagrant-ubuntu-trusty-64:~/DockerFiles/terraform$
```

2. terraform plan

```
vagrant@vagrant-ubuntu-trusty-64:~/DockerFiles/terraform$ terraform plan
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.

data.aws_iam_policy_document.ecs_task_execution_role: Refreshing state...

------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_ecs_cluster.nodecluster will be created
  + resource "aws_ecs_cluster" "nodecluster" {
      + arn  = (known after apply)
      + id   = (known after apply)
      + name = "white-hart"

      + setting {
          + name  = (known after apply)
          + value = (known after apply)
        }
    }

  # aws_ecs_service.main will be created
  + resource "aws_ecs_service" "main" {
      + cluster                            = "white-hart"
      + deployment_maximum_percent         = 200
      + deployment_minimum_healthy_percent = 100
      + enable_ecs_managed_tags            = false
      + iam_role                           = (known after apply)
      + id                                 = (known after apply)
      + launch_type                        = "FARGATE"
      + name                               = "service-ecs"
      + platform_version                   = (known after apply)
      + scheduling_strategy                = "REPLICA"
      + task_definition                    = (known after apply)
```

3. terraform apply

```
vagrant@vagrant-ubuntu-trusty-64:~/DockerFiles/terraform$ terraform apply
data.aws_iam_policy_document.ecs_task_execution_role: Refreshing state...

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_ecs_cluster.nodecluster will be created
  + resource "aws_ecs_cluster" "nodecluster" {
      + arn  = (known after apply)
      + id   = (known after apply)
      + name = "white-hart"

      + setting {
          + name  = (known after apply)
          + value = (known after apply)
        }
    }

  # aws_ecs_service.main will be created
  + resource "aws_ecs_service" "main" {
      + cluster                            = "white-hart"
      + deployment_maximum_percent         = 200
      + deployment_minimum_healthy_percent = 100
      + enable_ecs_managed_tags            = false
      + iam_role                           = (known after apply)
      + id                                 = (known after apply)
      + launch_type                        = "FARGATE"
      + name                               = "service-ecs"
      + platform_version                   = (known after apply)
      + scheduling_strategy                = "REPLICA"
      + task_definition                    = (known after apply)

      + network_configuration {
          + assign_public_ip = true
          + security_groups  = (known after apply)
          + subnets          = (known after apply)
        }
    }
```

```
Plan: 11 to add, 0 to change, 0 to destroy.

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

  Enter a value: yes

aws_ecs_cluster.nodecluster: Creating...
aws_vpc.main: Creating...
aws_iam_role.ecs_task_execution_role: Creating...
aws_iam_role.ecs_task_execution_role: Creation complete after 4s [id=MyEcsTaskExecutionRole]
aws_iam_role_policy_attachment.ecs_task_execution_role: Creating...
aws_ecs_task_definition.flaskapp: Creating...
aws_iam_role_policy_attachment.ecs_task_execution_role: Creation complete after 3s [id=MyEcsTaskExecutionRole-20201019195519298200000001]
aws_ecs_task_definition.flaskapp: Creation complete after 3s [id=service]
aws_ecs_cluster.nodecluster: Still creating... [10s elapsed]
aws_vpc.main: Still creating... [10s elapsed]
aws_ecs_cluster.nodecluster: Creation complete after 15s [id=arn:aws:ecs:us-east-1:900948362243:cluster/white-hart]
aws_vpc.main: Creation complete after 18s [id=vpc-0527b108df7d6fcbd]
aws_security_group.accessgroups: Creating...
aws_subnet.main[1]: Creating...
aws_internet_gateway.internetgateway: Creating...
aws_subnet.main[0]: Creating...
aws_subnet.main[0]: Creation complete after 6s [id=subnet-0b585400e07342bf5]
aws_subnet.main[1]: Creation complete after 6s [id=subnet-0ef45b134291c2bbd]
aws_internet_gateway.internetgateway: Creation complete after 8s [id=igw-0d38859669aabee11]
aws_route.internet_access: Creating...
aws_security_group.accessgroups: Still creating... [10s elapsed]
aws_security_group.accessgroups: Creation complete after 11s [id=sg-0a1cf2499946f909a]
aws_ecs_service.main: Creating...
aws_route.internet_access: Creation complete after 4s [id=r-rtb-0955643070caeb3991080289494]
aws_ecs_service.main: Creation complete after 4s [id=arn:aws:ecs:us-east-1:900948362243:service/service-ecs]

Apply complete! Resources: 11 added, 0 changed, 0 destroyed.
vagrant@vagrant-ubuntu-trusty-64:~/DockerFiles/terraform$
```

## Cluster on AWS