## Directions for Completing your Project

### A. AWS Steps

1. Log in to the AWS management console, as a **Root user**. Find and select the IAM (Identity and Access Management) service.
2. Click on "Group" menu item from the left sidebar. Create a new group and name it "jenkins", and attach the following policies:
   - AmazonEC2FullAccess
   - AmazonVPCFullAccess
   - AmazonS3FullAccess
3. Create an IAM user:
   - Click on "Users" menu item from the left sidebar. Create a new IAM User, select "Users" from the left sidebar, then "Add users", and use "jenkins" as the user name. Click on both "programmatic access" and "AWS management console access." The defaults for auto-generated password and "users must create a new password at next sign-in" are OK and should be kept. Hit "Next", and add the "jenkins" user to the "jenkins" group. Hit "next," no need to add "Tags." Review, and accept. Capture the Access Key, Secret Access Key, and the password so that you can log in as IAM user in the next step, easily (a pdf download the csv file).
   - **Screenshot#3** - Go to IAM dashboard, as the next user. Click on "Users" menu item from the sidebar and select the user that you have just created from the displayed result. Click on the "Permissions" tab. **Please provide a screenshot of the AWS console with the permissions being created. Name this 'screenshot#3'.**
   - Copy the IAM User sign in link from the IAM Dashboard.
4. Sign in as the new IAM user in a new browser window.
   - First, sign out of the AWS console. Then, use the IAM User sign-in link copied from the previous step.
   - Alternatively, you can login as the root user. Go to IAM dashboard. Click on "Users" menu item from the left sidebar. Select the 'jenkins' user link, and go to the Security credentials tab. Copy the "Console sign-in link".
   - If you haven't copied the IAM User sign-in link, you can generate the URL as follows:
     - In the following syntax `https://your_aws_account_id-signin.aws.amazon.com/console/`, replace the `your_aws_account_id` with your AWS account number without the hyphens (for example, if your AWS account number is 1234-5678-9012, your AWS account ID is 123456789012).
5. Create a key-key pair for access to your instances. Choose EC2 in the service after logging in. Select "Key Pairs" from the sidebar on the left, from the "Network and Security" section. Enter the "jenpeline" name when prompted. Save the "*.pem" file. If you will use an SSH client on a Mac or Linux computer to connect to your Linux instance, use the following command to set the permissions of your private key file so that only you can read it:

```
chmod 400 your_user_name-key-pair-region_name.pem
```

6. Launch the EC2 (t2.micro for the free-tier, pick "Ubuntu 18.04 LTS amalia," review, and when hitting "launch" ensure that an existing pair ("jenpeline") from before is selected. If you're not using the right key pair, you cannot log in. **Now, an Ubuntu 18.04 t2.micro instance is launched in the AWS EC2, that can be accessed via SSH using the PEM file. A screenshot showing the unique AWS URL of your EC2 instance should be captured. Name this 'screenshot#4.'**
7. Once launched, create a security group for this VM. In the left sidebar, under Network and Security, select "Security Groups." Under name, use "jenkins", description "Sans Jenkins security group." VPC should have the default one used. Click Add Rule. Custom TCP Rule, Protocol: TCP, Port Range: 8080, Source: 0.0.0.0/0 Then add the SSH rule. Protocol: SSH, Port range: 22, From anywhere, Source: 0.0.0.0/0 Then hit "create."
8. Go back to instances, and right click the running instance, select Networking and change the security groups. Select the jenkins security group that was created previously.
9. To connect to your instance using your key pair. Follow these steps. Right click your running instance and select "Connect." Then follow the instructions to SSH into it. For example, here are the directions that popped up in my AWS console on how to SSH into my machine (your directions will not look exactly like this, since portions of the information, like the full domain address of the instance, will be different):

```
Find the pem file, ensure
it has the right permissions (if on Mac or Linux)

    chmod 400 pipeline.pem

if on Mac or Linux then open the terminal and ssh into it:

    ssh -i "pipeline.pem"
    ubuntu@ec2-18-212-139-54.compute-1.amazonaws.com

if on Windows, an SSH client will need to be installed (PuTTy), follow
the guide at:

    https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/get-set-up-for
    -amazon-ec2.html#putty-or-kitty-ssh.
```

### B. Install Jenkins On Ubuntu

Here are the key commands for installation:

- `apt update`
- `apt upgrade`
- `apt install default-jdk`

The Jenkins version you get with the default Ubuntu packages is often not the latest available version: you can get from the Jenkins project itself. For the most recent features and fixes, you can use the packages from the Jenkins site to install Jenkins.

First, use `wget` to add the repo key to the system:

```
wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
```

When the key is added, the system returns OK. Next, append the Debian package repo address to the server's sources.list:

```
/pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
```

When both of these are all set, run `update` so that `apt` will use the new repo:

```
sudo apt update
```

Lastly, install Jenkins and its dependencies:

```
sudo apt install jenkins
```

Since `systemctl` doesn't produce output, you can use its `status` command to confirm that Jenkins began successfully:

```
sudo systemctl status jenkins
```

If all went well, the first lines of the output will show that the service is active and configured to start at boot, as shown below:

```
● jenkins.service - LSB: Start Jenkins at boot time
   Loaded: loaded (/etc/init.d/jenkins; generated)
   Active: active (exited) since Mon 2019-05-19 11:23:00 UTC; 1min ago
     Docs: man:systemd-sysv-generator(8)
  Tasks: 0 (limit: 1152)
  CGroup: /system.slice/jenkins.service
```

### C. Set Up Jenkins

1. Visit Jenkins on its default port, 8080, with your server IP address or domain name included like this: http://your_server_ip_or_domain:8080.
2. **Please provide a screenshot that includes the unique AWS url (ensure the FQDN is used, not the IP). Name this "screenshot#5."**
3. Next you will see the "Unlock Jenkins" screen, displaying the location of the initial password. In the terminal, use `cat` to show the password:

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

4. Copy and paste the 32-character alphanumeric password from the terminal into the Admin password field, then Continue.
5. The next screen gives you the choice of installing recommended plugins, or selecting specific plugins. I choose the install suggested plugins option, which quickly begins the installation process.
6. When installation is complete, you are prompted to set up the first admin user. Create the admin user and make note of both the user and password to use in the future.
7. You next see an Instance Configuration page, asking you to confirm the preferred URL for your Jenkins instance. Confirm the address, click save and finish.

### D. Install Blue Ocean plugin

1. "Blue Ocean" and other required plugins need to be installed. Logged in as an admin, go to the top-left, click 'Jenkins', then 'manage Jenkins', and select 'Manage Plugins'.
2. Use the "Available" tab, filter by "Blue Ocean," select the first option ("BlueOcean aggregator") and install without a restart.
3. Filter once again for "pipeline-aws" and install, this time selecting "Download now and install after restart."
4. Once all plugins are installed, Jenkins will restart. If it hasn't restarted, run the following in the VM:

```
sudo systemctl restart jenkins
```

5. Verify everything is working for Blue Ocean by logging in. An "Open Blue Ocean" link should show up in the sidebar. Click it, and it will take you to the "Blue Ocean" screen, where we will have to add a project.
6. **Please provide a screenshot that includes the unique AWS url, and shows the sidebar with the Blue Ocean link (ensure the FQDN is used, not the IP). Name this "screenshot#6."**
7. A welcome screen will appear, telling you it is time to create your first pipeline.
8. Click "create pipeline."

### E. Set up a GitHub Repository

**Note: A GitHub account is required for the next steps.**

1. Create a new repository in your GitHub account named 'static'. In the repo, create two files: "Index.html" and "jenkinsfile".
   - The "jenkinsfile" should look like this:

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                sh "echo "Hello World""
                sh '''
                    echo "Multiline shell steps works too"
                    ls -lrtmh
                '''
            }
        }
    }
}
```

   - The "Index.html" file should look like this:

```
<!doctype html>
<html>
  <head>
    <title>Static HTML Site</title>
  </head>
  <body>
    <p>This is the first paragraph in a simple static HTML site. There is no s
  </body>
</html>
```

2. Commit and push your changes.
3. Select GitHub from the options available, a token needs to be generated. A link to https://github.com/settings/tokens/new?scopes=repo,read:user,user:email,write:repo_hook needs to be clicked to generate a token for Jenkins to use. You can select the default scopes in the opened link, that defines the access for a personal token for Jenkins.
4. Authenticate in GitHub, and add a note for what this token is cause for later removal("Jenkins-Pipeline."
5. Make sure you copy the token - there is no way to see it again!
6. After pasting the token into the form in Jenkins, click "connect", and your account should show up. If your account belongs to multiple organizations, they will be listed - make sure you use your personal account and organization.
7. Next, search for "static" so that the repo is matched, and click "create pipeline."
8. The pipeline should show up with a never run.
9. **Please provide a screenshot that includes the unique AWS url, and shows the GitHub project as a pipeline (ensure the FQDN is used, not the IP). Name this "screenshot#7."**
10. In the page where the 'static' job shows, there is a gear icon - click on it to edit the job directly. Find the "Scan repository triggers" and click on "Periodically if not otherwise ran," and select an interval of 1 minutes.

**This completes the initial pipeline setup for Jenkins.**

### F. Set up AWS credentials in Jenkins

Credentials need to be created so that they can be used in our pipeline.

1. Leave the Blue Ocean UI, and go back to the main Jenkins page. Then click on the "Credentials" link from the sidebar.
2. Click on "global" from the list, and then "Add credentials" from the sidebar.
3. Choose "AWS Credentials" from the dropdown, add "aws-static" on ID, add a description like "Static HTML pipeline in AWS," and fill in the AWS Key and Secret Access Key generated where the IAM role was created.
4. Click OK, and the credentials should now be available for the rest of the system.

### G. Set up S3 Bucket

In order to publish the contents, we need to create a bucket in S3 that has the right permissions to add files and that others can get access to.

1. Log in to the console for the 'jenkins' user that was created in the beginning, and select the 'S3' service.
2. Create a new bucket, and make it a unique name. The system will not let you continue creation if the name is not unique. Remember this name for later configuration.
3. Take note of the region. This will also be used later.
4. Select "next" to configure options, do not select any special options, then hit "next" again.
5. For "Set Permissions," uncheck the "Block all public access." Hit "next" once again, review the settings, and click "continue."
6. The bucket should be available in the S3 console. Click it to get to the configuration panel for that bucket.
7. Select the "Properties" tab, and click on "Static website hosting." Enable the "Use this bucket to host a website" and type in "index.html" for the index document. Click "save."
8. Select the "Permissions" tab.
9. Click on "Bucket policy" and add the following:

```
            {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Sid": "AllUsersGetObject",
                        "Effect": "Allow",
                        "Principal": "*",
                        "Action": "s3:GetObject",
                        "Resource": "arn:aws:s3:::NAME_OF_BUCKET/*"
                    }
                ]
            }
```

10. Replace 'NAME_OF_BUCKET' with the bucket that was just created.
11. Click save, and "Permissions policy" should now show in the UI.

### H. Set up pipeline for AWS

1. Go back to your project, and edit "jenkinsfile." Replace the "Build" stage with "Upload to AWS."
2. The stage should upload the "index.html" file using the region and credentials for AWS.
3. The "s4ansible" and "s3upload" actions from the 'pipeline-aws' plugin should be used. "s4ansible" is documented here: https://github.com/jenkins-ci/pipeline-aws-plugin#s4ansible, "s3upload" is documented here: https://github.com/jenkins-ci/pipeline-aws-plugin#s3upload.

**The "credentials" need to match the name given when they were created in Jenkins (see the "Set up AWS credentials in Jenkins" section on this page for reference).**

4. Save, commit, and push. Within minutes, a new run should appear - it should be successful.
5. To verify, go to the URL where the static S3 website is: http://BUCKET_NAME.s3-website-REGION.amazonaws.com/. Replace "BUCKET_NAME" with the bucket that was created early, and "REGION" with the according region where the bucket exists.

The contents of the "index.html" file should exist there.

6. **Please provide a screenshot that includes the unique AWS url, and shows "index.html" rendered. Name it "screenshot#8."**

### I. Add another stage in pipeline

The index file has an invalid HTML in there. To prevent getting an invalid HTML, we are going to run a linter so that it fails the job if anything gets in that is invalid.

1. Connect to the host where Jenkins is installed, and install the following system dependency: `sudo apt-get install -y tidy` This would install the `tidy` linter in the server.
2. Now that the system dependency is installed, add a new stage for linting the HTML in the project, before the "Upload to AWS" stage. that runs the 'tidy' utility. Name it "Lint HTML." The command that runs the linter is `tidy -q -e *.html`.
3. Commit this new change and push to GitHub, then wait a couple of minutes. ***The build should now FAIL at the linting stage, because the HTML is invalid and has an error.***
4. **Please provide a screenshot that includes the unique AWS url, and shows the failure when linting. Name it "screenshot#9."**
5. Go edit the "index.html" file again, find the invalid portion, and replace it with the correct version so that the linter does not complain. The job should now pass without a problems.
6. **Please provide a screenshot that includes the unique AWS url, and shows passing the linting stage and deploys to S3. Name it "screenshot#10."**

### This is what the last passing build should look like

Please note the "Upload complete" message, and the "Lint HTML" step showing as the first item after "Stats."