Github URL: <https://github.com/Radhika-cloud/devops-mindtree-20191115>

**Jenkins-stage-docker-terraform-ansible-file**

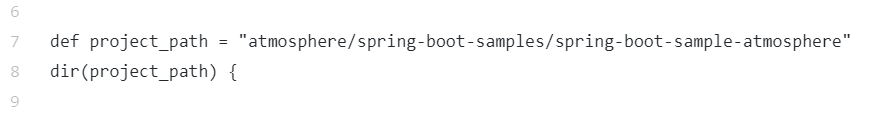
**Jenkinsfile**: <https://github.com/Radhika-cloud/devops-mindtree-20191115/blob/master/Jenkins/Jenkins-stage-docker-terraform-ansible-file.save>

**Get source code from repo:**

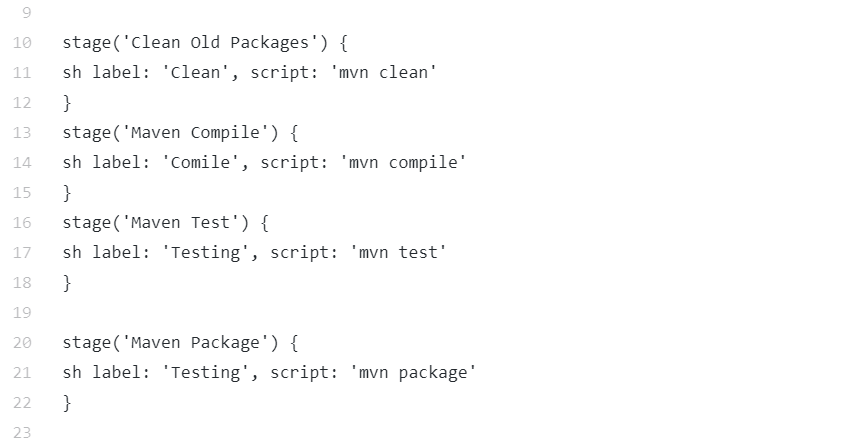
**Node**:node is Scripted Pipeline-specific syntax that instructs Jenkins to execute this Pipeline (and any stages contained within it), on any available agent/node. Jenkins will schedule and run all the steps once any node is available and creates a specific workspace directory.

**Stage**: stage is a block that contains a series of steps. A stage block can be named anything; it is used to visualize the pipeline process.

**Set project path:**



The Git repo has been copied and project path is set. ‘dir(project\_path)’ sets the working directory for running all the commands from the Jenkinsfile.

**Running maven commands:**

**mvn clean**: it clears out the existing classes that you compiled from last compile

**mvn compile:** Compiles the source code of the project.

Software that converts a Java source program into bytecode (intermediate language) that converts bytecode into machine language. It may also refer to compiling the source code into the native language of a particular hardware platform.

**mvn test**: Runs the tests against the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed.

**mvn package**: Builds the project and packages it in its distributable format, resulting JAR file into the target directory.

**File archive:**

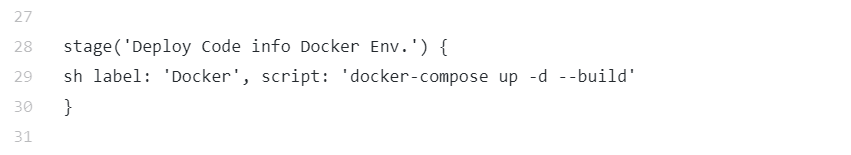


This stage is to save files “outside build workspace” to get after others builds.

The jar file is important for deploy. And after other build, this file is replaced or can be removed, in order to avoid it, we need archive the jar file.

‘target/\*.jar’ is to save multiple files from a specific folder.

**Deploying to the Docker environment:**



This command will start up our application.

Docker-compose is use to run the multiple containers as a single service.

With Compose, we use a YAML file to configure our application's services. Then, with the above single command, we create and start all the services from our configuration.

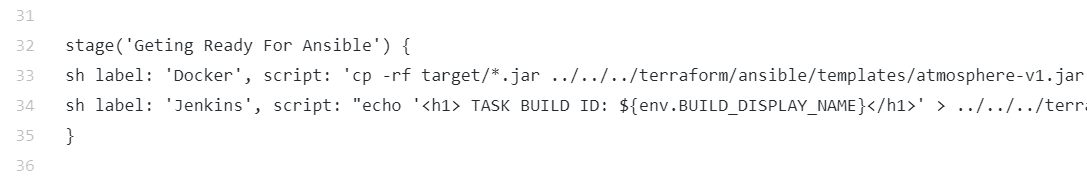
In this command:

-d (--detach): run containers in background

--build: build images before creating containers.

It referring to which container?

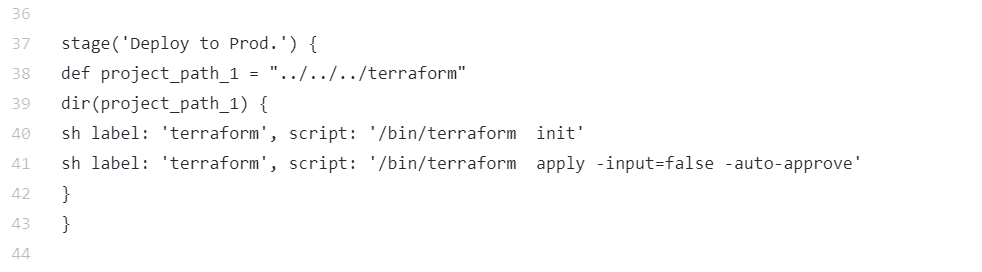
**Ansible and Terraform:**



In above command,

-r option will copy the files and directory recursively.

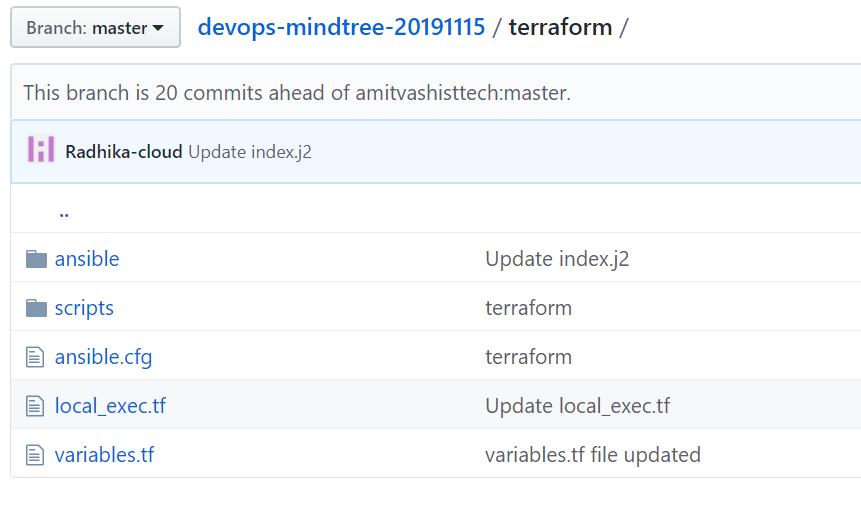
-f will copy the files and directory forcefully. (For some scenarios where existing destination file cannot be opened and removed we use -f)



The ‘**terraform init’** command is used to initialize a working directory containing Terraform configuration files. By default, ‘**terraform init’** assumes that the working directory already contains a configuration and will attempt to initialize that configuration. And here in this project, **local\_exec.tf** is the configuration file.

**Terraform**

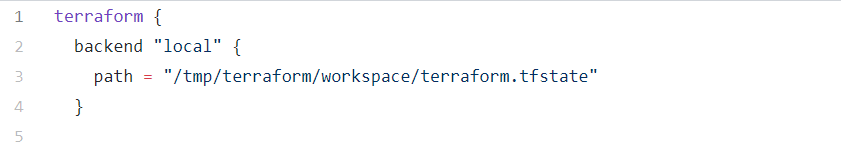
**Folder structure:**



**variables.tf:** This file describes all the variables that are used within the local\_exec.tf file.

For ex: pvt\_key, sg-id, region etc.

**local\_exec.tf:** This file contains the template that describes all the resources that need to be deployed. This is agnostic to the cloud account and region.



This terraform.tfstate contains the state of our cloud infrastructure after deployment. This file is referenced when modifying/destroying the resources.

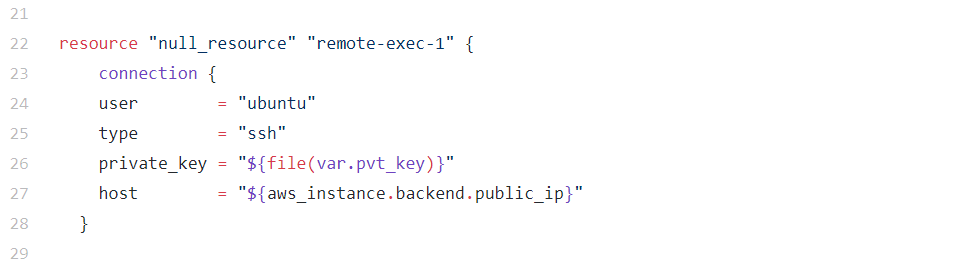


The Amazon Web Services (AWS) provider is used to interact with the many resources supported by AWS. The provider needs to be configured with the proper credentials before it can be used.



Here we are creating a server with name “backend” with instance type t2.micro. AMI is used to create a virtual machine within the Amazon Elastic Compute Cloud ("EC2").

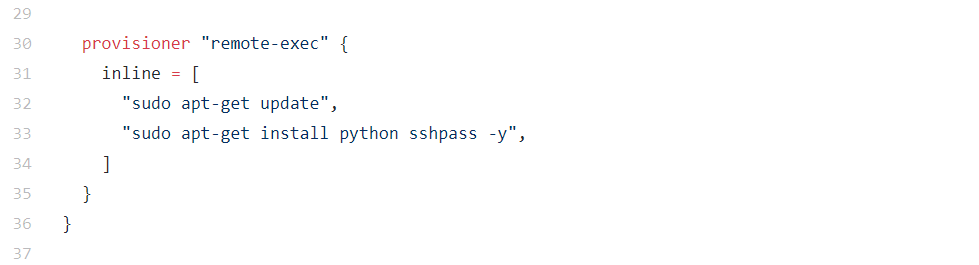
Key name and sg-id are mentioned in variables.tf file.



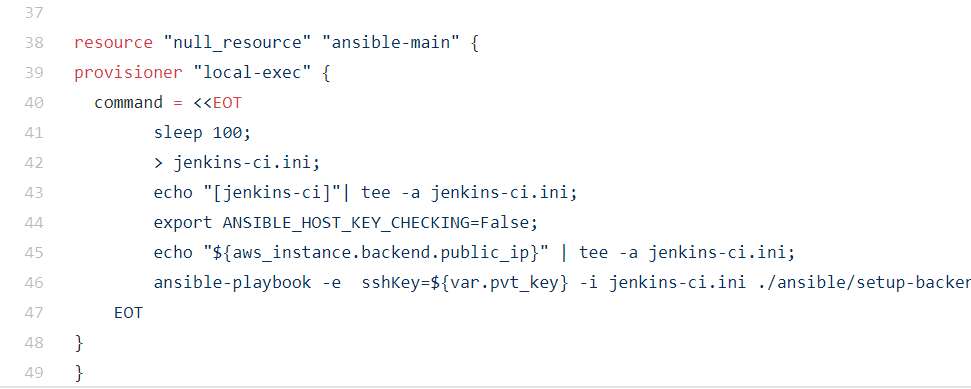
This generates the template internally, that means the data is available, but not dumped anywhere.

We use “null-resource”, for this purpose so the generation of template is independent of any other executing resource.

Terraform apply

Ansible requires Python to be installed on the remote machine as well as the local machine.

"sudo apt-get install python sshpass -y",



This is where we configure the instance with ansible-playbook

The sleep command is used to delay for a specified amount of time, here we are delaying for 100 seconds to give the instance some time to boot up.

EOT- End-of-Transmission,

*ANSIBLE\_HOST\_KEY\_CHECKING* environment variable to False, to make Ansible trust the newly-launched server.

Then we are running the ansible-playbook command, with a specified private key, and a specified Ansible inventory (./ansible/setup-backend.yaml)