**Jenkins: assignment**

**1. Prepare tools and repositories for the module.**

* create a git repository in Github/Gitlab/any. Let's name it "codeRepo". You are going to store the source code and unit tests there. [Here](https://github.com/wardviaene/docker-demo) or [here](/marat_rysbekov/mp2019/blob/master/aws_jenkins/codeRepo/README.md) you can find (just clone it) the code, tests and Dockerfile for building your application docker image. Please watch Jenkins video course for more details on what is the code and what are those tests about.
* create a git repository in Github/Gitlab/any. Let's name it "cdRepo". This repository you are going to use for storing your Jenkinsfile, ansible playbook and Docker file.
* create a docker hub registry to store your artifacts (your application's built image)

**2. Build your own Jenkins docker image based on Jenkins latest docker image.**

Your custom image should be provisioned with next tools:

* docker
* pip
* ansible Please watch the video course where you can find details on how to build your custom Jenkins image.

**3. Run Jenkins.**

Run EC2 instance (you can do it manually). Run Jenkins as a docker container using your custom image which you built on the previous step. Be sure that you:

* exposed an appropriate ports for Jenkins container
* provided your Jenkins container with an access to host's docker api using docker socket
* mapped Jenkins\_home to the host's filesystem
* created a user to manage Jenkins
* installed suggested plugins for Jenkins on startup via wizard.

**4. Build your scripted ci/cd pipeline.**

Your pipeline should be parametrized and consist of several steps described below. Parameters for pipeline:

* branch name of the source code repository to perform a checkout for the code

**4.1 "Preparation" step.**

* clean the workspace
* checkout your codeRepo according to a branch or tag

**4.2 "Unit tests" step.**

* using nodejs docker image run unit tests against your code inside nodejs container using "npm test"
* using nodejs docker image build your code inside node js container using "npm install"

**4.3 "Buld" step.**

* ensure you stored docker registry credentials in Jenkins with "docker-hub" ID assigned to them
* build docker image with the Dockerfile provided within codeRepo
* using your docker hub credentials push the image into your docker registry. Docker image tag should contain the jenkins build number

**4.4 "Build infrastructure" step.**

Checkout the cdRepo where you should store the cloudformation template described below. Develop a Cloudformation template which includes the following AWS components:

* VPC (with "EnableDnsSupport" and "EnableDnsHostnames" options enabled);
* non-default route table;
* two subnets associated with the route table (with "MapPublicIpOnLaunch" option enabled);
* Internet gateway attached to the VPC;
* default route in the route table pointing to the Internet gateway.
* Security group allowing all incoming connections from your IP address, IP address of Karaganda EPAM office, public IP address of the Jenkins master and also allowing incoming HTTP connections from anywhere.
* EC2 instance which must be based on Amazon Linux 2 AMI and have a User Data script where you provide the host with public key to be able to provision the host via ansible.

The template must have the following input parameters:

* CIDR block for VPC (a default value must be pre-defined);
* CIDR block for subnet (a default value must be pre-defined);
* EC2 instance class (a list of allowed values must be pre-defined, "t2.micro" being the default);
* EC2 key pair name;
* Public IP address of the Jenkins master

The template must have a mapping to choose AMI image ID based on the AWS region.

The template must have the following outputs:

* Public IP address of the instance provisioned

NOTE: provide Jenkins master with aws credentials (aws secret id and key id) to let him provision AWS resources.

**4.5 "Deliver" step.**

Checkout the cdRepo where you should store the Ansible playbook described below. Develop an Ansible playbook that includes the following AWS provision tasks:

* install docker with all the dependencies needed
* pull the image you built in previous step
* run the image the way your application should be available via http on the provisioned host
* ensure that you have started the proper image tag

NOTE: provide Jenkins master with SSH credentials (private key) to let him provision the target host.

**4.6 "e2e test" step.**

Simple step checking that your application is available via http and showing that it works. This can be a simple curl call and it's output in a representative look.

**5. Provide your mentor with results.**

Please copy $Jenkins\_home/plugins and $Jenkins\_home/jobs folders and share it with your mentor. Please share a content of codeRepo and cdRepo with your mentor.

**Expected results:**

Build Jenkins pipeline in which described steps must be implemented. To verify your solution you can follow the next steps:

1. Create a new branch in your codeRepo. For example "dev" branch or any other name.
2. Change/brake your code and commit it into your newly created branch.
3. Run your pipeline and make it checkout your code from the newly created branch
4. Your pipeline should build the code that you changed in previous step, build docker image with an appropriate tag (build number), push it into registry and then deploy this image into your prepared AWS environment and start application with the new code inside.
5. Pipeline output must show the result of the e2e step and you can see if your new setup is broken/changed.