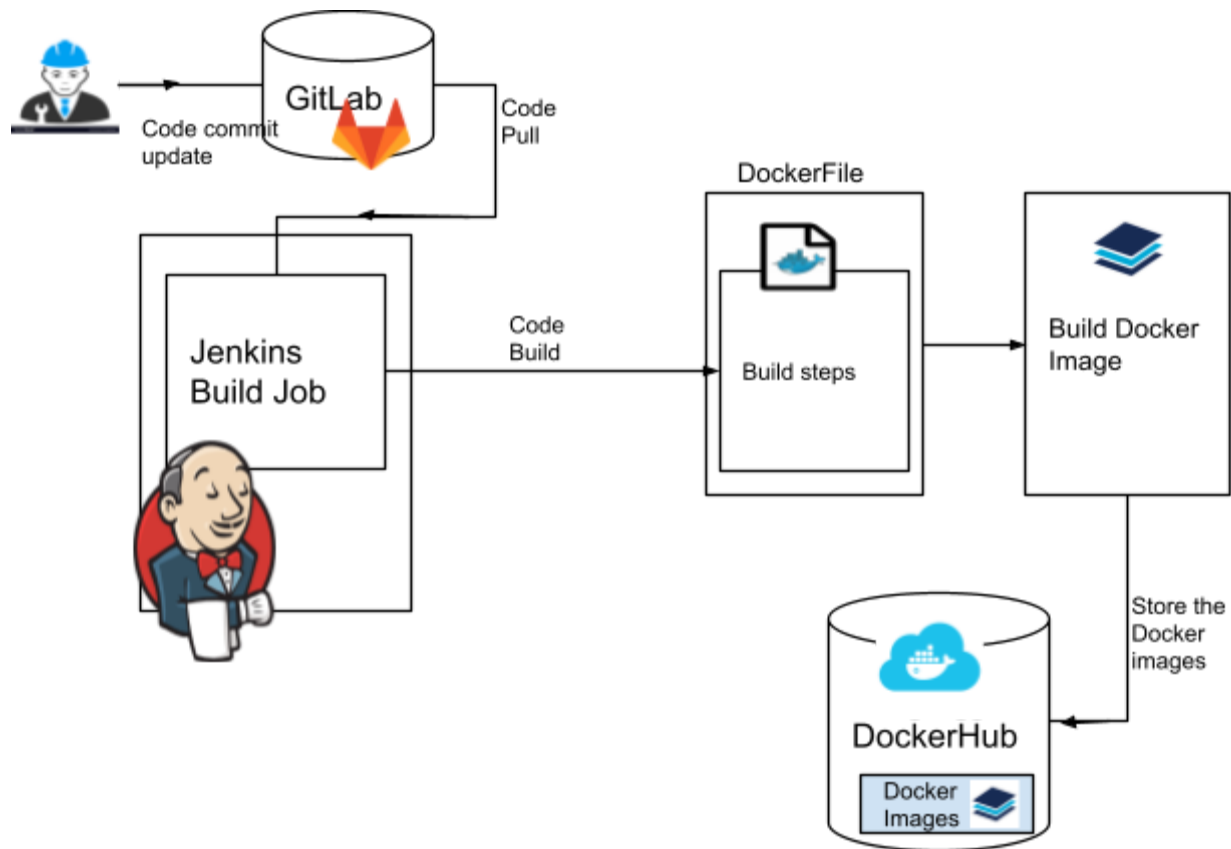


Build and deployment Jobs Configuration

Application Build Process



The developer makes a request to do build/deploy after pushing the latest changes to the GitLab.

Developers can select respective branches to build the code.

As per the developer request, we need to do build and deployment to reflect the changes in application. We have two separate jobs for application build and Deployment.


Build job configuration:


Docker FT environment


All +					
S	W	Name ↓	Last Success	Last Failure	Last Duration
		build	N/A	N/A	N/A
		deploy	N/A	N/A	N/A


Go to build job and choose an environment. And click on any servers in the pre-prod environment for instance.


Jenkins > DOCKER > build > pre-prod > kal-agent >


 Up


 Status


 Changes


 Workspace


 Build with Parameters


 Delete Project

 Configure

 Rebuild Last


 Authorization


 Move

 Rename


Project kal-agent

Full project name: DOCKER/build/pre-prod/kal-agent



 [Workspace](#)

 [Recent Changes](#)

Permalinks

 **Build History** [trend](#)

x

 [RSS for all](#)  [RSS for failures](#)

Click on configure tab to see how the build job is configured.

The screenshot shows the 'General' tab of a build configuration interface. At the top, there are tabs: 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build', and 'Post-build Actions'. Under the 'General' tab, there are three checkboxes: 'Disable Rebuilding for this job' (unchecked), 'This build requires lockable resources' (unchecked), and 'This project is parameterised' (checked). Below these, there is a 'String Parameter' section. It contains a 'Name' field with the value 'image_tag', a 'Default Value' field (empty), and a 'Description' field with the text 'Enter The image_tag to build'. There is a '[Plain text] Preview' link and a 'Trim the string' checkbox (unchecked). At the bottom of the parameter section is an 'Add Parameter' button with a dropdown arrow. A red 'X' icon is visible in the top right corner of the parameter section.

Under the General tab section, we can see the job is parameterized. By this we are asked to give Image_tag at the time of build.

The screenshot shows the 'Source Code Management' tab of the same build configuration interface. At the top, there are tabs: 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build', and 'Post-build Actions'. The 'Source Code Management' tab is selected. It contains a 'Source Code Management' section with two radio buttons: 'None' (unchecked) and 'Git' (checked). Below this, there is a 'Repositories' section. It contains a 'Repository URL' field with the value 'https://gitlab.com/kaleidofin/kaleidofin-agent-ui.git', a 'Credentials' dropdown menu with the value 'prabhakaran/***** (Prabhakaran-GitLab)', and an 'Add' button. There is an 'Advanced...' button and an 'Add Repository' button. Below the 'Repositories' section is a 'Branches to build' section. It contains a 'Branch Specifier (blank for \'any\')' field with the value '*/uat' and an 'Add Branch' button. Below the 'Branches to build' section is a 'Repository browser' dropdown menu with the value '(Auto)'. At the bottom, there is an 'Additional Behaviours' section with an 'Add' button. At the very bottom, there are two radio buttons: 'Multiple SCMs' (unchecked) and 'Subversion' (unchecked).

Three things to apply here in SCM(Git):

1. code repo for this service
2. Credentials to pull the code from GitLab
3. pull the code with branch specific

Under the Build environment tab, we will choose to delete the workspace before the build starts and set the build name to be displayed as the **image_tag** given. This can be seen in build history status after the build job is completed.

The screenshot shows the Jenkins configuration interface for a build job, specifically the 'Build Environment' tab. The tabs at the top are 'General', 'Source Code Management', 'Build Triggers', 'Build Environment' (selected), 'Build', and 'Post-build Actions'. The 'Build Environment' section has a title 'Build Environment' and a list of options, each with a checkbox and a help icon. The first option, 'Delete workspace before build starts', is checked. Below this is an 'Advanced...' button. The 'Set Build Name' option at the bottom is also checked, and its corresponding text field contains the value '\${image_tag}'. Another 'Advanced...' button is located at the bottom right of the configuration area.

General Source Code Management Build Triggers **Build Environment** Build Post-build Actions

Build Environment

☒ Delete workspace before build starts

Advanced...

- ☐ Use secret text(s) or file(s) ?
- ☐ Provide Configuration files ?
- ☐ Send files or execute commands over SSH before the build starts ?
- ☐ Send files or execute commands over SSH after the build runs ?
- ☐ Abort the build if it's stuck
- ☐ Add timestamps to the Console Output
- ☐ Ant/Ivy-Artifactory Integration
- ☐ Generic-Artifactory Integration
- ☐ Gradle-Artifactory Integration
- ☐ Inject environment variables to the build process ?
- ☐ Inject passwords to the build as environment variables
- ☐ Inspect build log for published Gradle build scans
- ☐ Maven3-Artifactory Integration
- ☐ Prepare SonarQube Scanner environment ?
- ☐ Provide Node & npm bin/ folder to PATH
- ☒ Set Build Name ?

Build Name

Advanced...

Under Build section, we will give shell commands to generate the artifact and build docker image, push the images to Dockerhub

Build

Execute shell

Command

```
cd kaleidofin-agent-ui ; npm install; npm rebuild node-sass; yarn run build ;
```

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

Advanced...

Execute shell

Command

```
cd kaleidofin-agent-ui/  
echo " Create a docker image"  
docker build -t kaleidofin/kfin:${image_tag} .  
echo " Push the docker images to the Hub"  
docker push kaleidofin/kfin:${image_tag}
```

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

Advanced...

Save

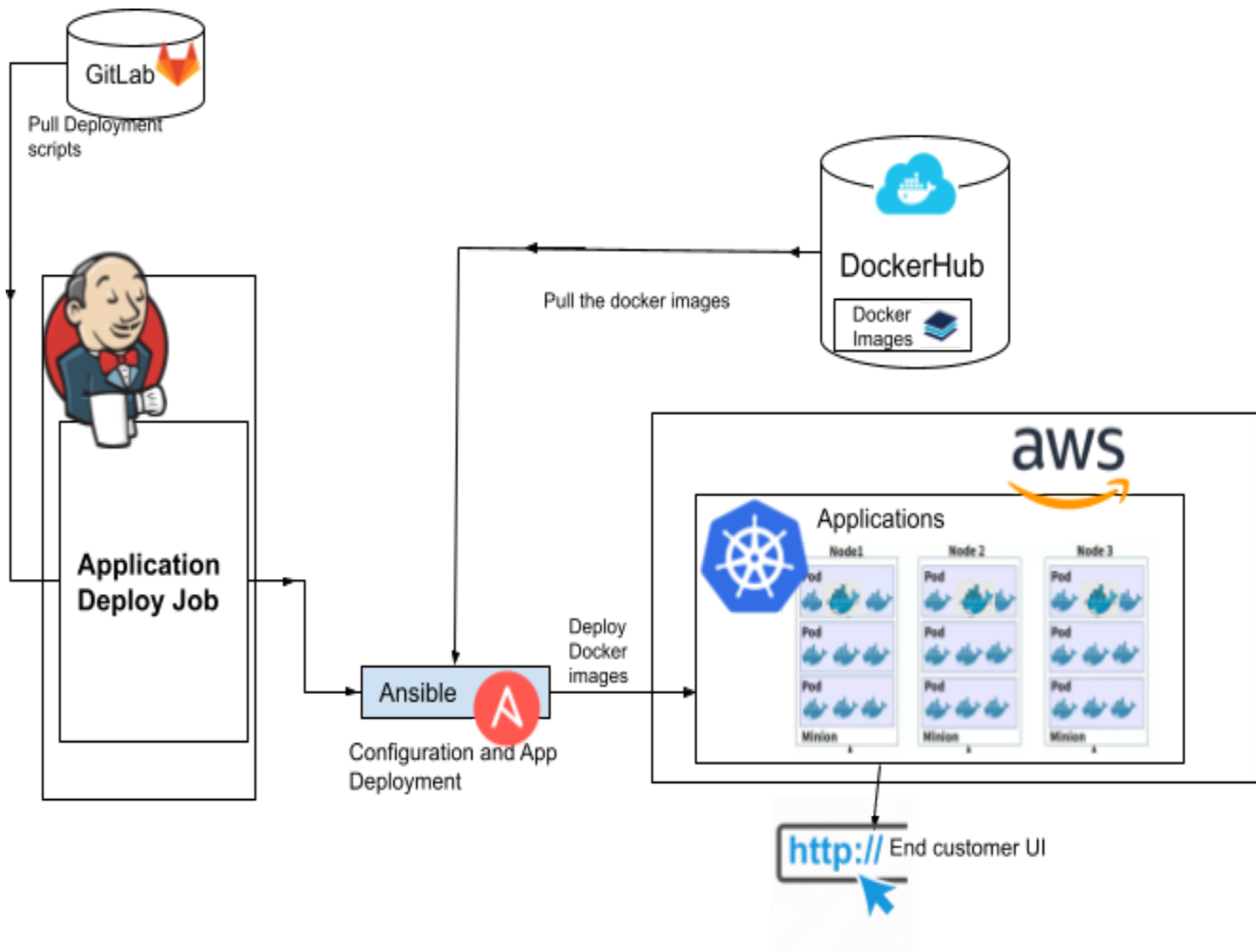
Apply

The commands in the first section, used to install any dependencies and generate an artifact which can be used to build an image.

The commands in the second section, creates a docker image using the artifact generated. Then the image built is pushed to docker hub. The dockerhub logins are configured with jenkins.

This is how the build job is configured



Deployment Process




Jenkins and Ansible are running on the same machine (EC2). All the application deployments are deployed into EKS k8s cluster which is managed by AWS.


Deploy Job Configuration:

Now go to the Deploy job and choose any environment. Click on any server to deploy the job. For demo, we are doing the deployment in a kal-agent server.

-  Up
-  Status
-  Changes
-  Delete Pipeline
-  Configure
-  Authorization
-  Move
-  Full Stage View
-  Rename
-  Pipeline Syntax

 **Build History** [trend](#) 



 [RSS for all](#)  [RSS for failures](#)

Pipeline kal-agent

Full project name: DOCKER/deploy/dev/kal-agent



[Recent Changes](#)

Stage View

No data available. This Pipeline has not yet run.

Permalinks

Click on the configure button to see how the deploy job is configured.

GeneralBuild TriggersAdvanced Project OptionsPipeline

☐ Pipeline speed/durability override

☐ Preserve stashes from completed builds

Rebuild options: ☐ Rebuild Without Asking For Parameters

☐ Disable Rebuilding for this job

☒ This project is parameterised

String Parameter

Nameimage_tag

Default Value

DescriptionEnter image_tag

[Plain text] [Preview](#)

☐ Trim the string

Choice Parameter

NameEnvironment

Choicesft1ft2developmentuat

DescriptionEnter the Environment

[Plain text] [Preview](#)

Choice Parameter

business_group

Choiceskaleidofin-webkal-ft

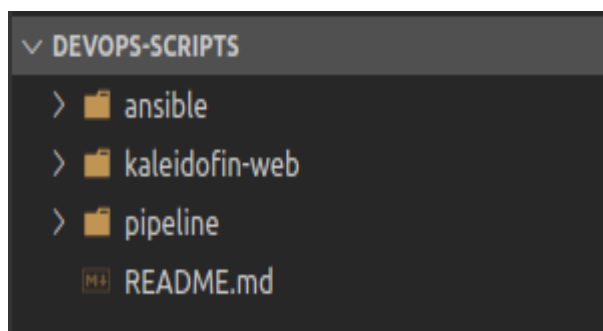
DescriptionEnter the business_group

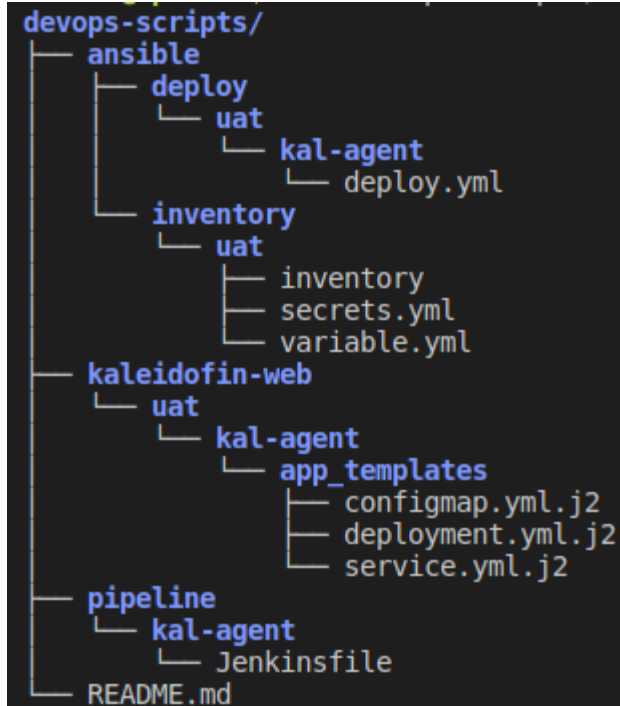
Under the General tab, we have two parameters defined. The string parameter is to give a name to image_tag and choice parameter is to choose the environments.

Under pipeline section, we have a jenkins pipeline script,

The screenshot shows the Jenkins Pipeline configuration interface. At the top, there are tabs for 'General', 'Build Triggers', 'Advanced Project Options', and 'Pipeline'. The 'Pipeline' tab is selected. Below the tabs, the 'Definition' section is set to 'Pipeline script from SCM'. The 'SCM' is configured as 'Git'. Under 'Repositories', the 'Repository URL' is 'https://gitlab.com/kaleidofin/devops-scripts.git' and the 'Credentials' are 'prabhakaranang/* (Prabhakaran-GitLab)'. The 'Branches to build' section has a 'Branch Specifier (blank for \'any\')' set to '*/development'. The 'Repository browser' is set to '(Auto)'. There is an 'Add' button for 'Additional Behaviours'. The 'Script Path' is 'pipeline/kal-agent/Jenkinsfile'. The 'Lightweight checkout' checkbox is checked. At the bottom left, there are 'Save' and 'Apply' buttons. A 'Pipeline Syntax' link is also visible.

All the yaml files are found in the development branch of <https://gitlab.com/kaleidofin/devops-scripts.git>





```

pipeline {
    agent any

    parameters {
        string(name: 'image_tag', defaultValue: '', description: 'Enter image_tag')
        choice(name: 'Environment', choices: ['ft1','ft2','development','uat'], description: 'Enter the Environment')
        choice(name: 'business_group', choices: ['kaleidofin-web','kal-ft'], description: 'Enter the business_group')
    }

    stages {
        stage('Deployment to $env k8s cluster') {
            steps {
                buildName "${params.image_tag}"
                sh "ansible-playbook -i ansible/inventory/${params.Environment}/
                ansible/deploy/${params.Environment}/${env.JOB_BASE_NAME}/deploy.yml
                --vault-password-file /var/lib/jenkins/secrets/vault-pass
                -extra-vars 'env=${params.Environment} business_group=${params.business_group} image_tag=${params.image_tag} ws=$WORKSPACE'"
            }
        }
    }
}
  
```

The above snippet is the jenkins pipeline script.

This pipeline is the Entrypoint to k8s deployment. So, here in this stage ansible playbook calls the deploy.yaml[1] by passing extra parameters[2] at runtime with variables[3]

1. Calls the kubectl command to run the deployment files
2. extra params like: env, business_groups, image_tag and ws
3. inventory file where ansible runs with credentials.

The inventory file is located at path: ansible/inventory/uat/inventory

```
ansible > inventory > development > inventory
1  [local]
2  localhost ansible_host=localhost ansible_ssh_user=deployer ansible_ssh_pass=devops136@@
3
4  [masters]
5  localhost
6
7  [all:vars]
8  #ansible_ssh_user=optit
9  ansible_python_interpreter=/usr/bin/python3
10 ansible_host_key_checking=False
11
12 #[defaults]
13 #host_key_checking = False
```

The `deploy.yaml` file is located at `ansible/deploy/uat/kal-agent/deploy.yaml`

```
ansible > deploy > development > kal-agent > yes deploy.yml > ...
```

```

1  ---
2  - hosts: masters
3    become_user: deployer
4
5
6  vars:
7    env: ft1
8    image_tag: dev-1.1.2
9    application: kal-agent
10   business_group: kaleidofini-web
11
12  vars_files:
13   - "{{ inventory_dir }}/secrets.yml"
14   - "{{ inventory_dir }}/variable.yml"
15

```

```

16 tasks:
17
18 - name: Create a directory
19   command: "mkdir -p ~/kube-deploy/{{ business_group }}/{{ env }}/{{ application }}"
20
21 - name: Copy the deployment file
22   copy:
23     src: "{{ ws }}/{{ business_group }}/{{ env }}/{{ application }}/"
24     dest: "~/kube-deploy/{{ business_group }}/{{ env }}/{{ application }}/"
25     force: yes
26
27 - name: "Copy the latest {{ application }} image {{ image_tag }} deploy"
28   template: src={{ item }} dest=~ / kube-deploy/{{ business_group }}/{{ env }}/{{ application }}/{{ item | basename |
29   regex_replace('.j2','') }}" force=yes
30   with_fileglob:
31     - "{{ ws }}/{{ business_group }}/{{ env }}/{{ application }}/app_templates/*.j2"
32
33 - name: "Creating {{ env }} namespace"
34   command: "kubectl create namespace {{ env }}"
35   ignore_errors: yes
36
37 - name: Deploying Dockerhub secrets
38   command: "kubectl create secret docker-registry dockersecrete --docker-server=https://index.docker.io/v1/
39   --docker-username={{ dockeruser }} --docker-password={{ dockerpwd }} --namespace={{ env }}"
40   no_log: True
41   ignore_errors: yes
42
43
44 - name: "Deploying {{ application }} Application"
45   command: "kubectl apply -f ~/kube-deploy/{{ business_group }}/{{ env }}/{{ application }}/ --namespace={{ env }}"
46   #ignore_errors: yes
47

```

In the above ansible file, we are executing the tasks as a deployer user and the k8s cluster configuration is also done to this user(/home/deployer/.kube/config) so that we can execute kubectl commands to create the deployments in k8s cluster.

The ansible vars will be replaced by the runtime variables while having --extra-vars in ansible-playbook

The secrets.yaml has sensitive data, so it is protected with ansible-vault . and we also given the password file path to it in the execution command.

The task process in deploy.yml consists of 3 parts:

1. Creating a working directory.
2. Having the latest devops code in the existing workspace.
3. Rendering jinja2 templates (passes values to variables and removes .j2)
4. Creating a namespace (ignores if already exists)
5. Deploying the infra

deployment.yml.j2

file:

```
kaleidofin-web > development > kal-agent > app_templates > deployment.yml.j2
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    creationTimestamp: null
5    labels:
6      run: kal-agent
7    name: kal-agent
8    namespace: {{ env }}
9  spec:
10   replicas: 1
11   selector:
12     matchLabels:
13       run: kal-agent
14   strategy: {}
15   template:
16     metadata:
17       creationTimestamp: null
18       labels:
19         run: kal-agent
20     spec:
21       containers:
22       - image: kaleidofin/kfin:{{ image_tag }}
23         name: kal-agent
24         imagePullPolicy: Always
25         #envFrom:
26         # - configMapRef:
27         #   name: kal-agent
28         ports:
29         - containerPort: 80
30
31       #resources:
32       #  requests:
33       #    cpu: "{{kal_agent_cpu_reserve}}"
34       #    memory: "{{kal_agent_memory_reserve}}"
35       #  limits:
36       #    cpu: "{{kal_agent_cpu_limit}}"
37       #    memory: "{{kal_agent_memory_limit}}" */
38
39     imagePullSecrets:
40     - name: dockersecrete
41
42   status: {}
```

As we are pulling the container image from the private docker hub repository, we should use k8s secrets here to pull the image from Dockerhub.

In the final task, deployment.yaml file with specified namespace is executed.

This is how the deployment job configuration is done.