**Paywallet**

**AWS environment configuration Guide**

**Table of Contents Page**

**PART – 1: CREATING RESOURCES IN AZURE**

* 1. **Architecture diagram of Pre-Prod environment 4**
  2. **Architecture diagram of Pre-Prod environment 5**
  3. **Architecture diagram of Pre-Prod environment 5**

**2. Create Resource group 6**

3. Create Aurora MYSQL Server And Aurora POSTGRESQL Databases 6

3.1. Create Aurora MySQL server for pre-prod environment with the name dev-fineract and uat-fineract 6

**3.2 Create Aurora PostgreSQL server for pre-prod environment with the name dev-kong-konga and uat-kong-konga-keycloak 7**

4. Create Bastion Server To Access To Create And Access Each Environment

**In AWS 9**

**5. Create EKS Cluster And Amazon Elastic Container Registry** **11**

**5.1 Install Terraform on Bastion. 11**

**5.2 Install AWS CLI on Bastion and configure the aws configuration. 11**

**5.3 Terraform script to create EKS cluster from Ubuntu. 11**

* 1. **Create the ECR Registry 18**

6. Accessing EKS Cluster From UBUNTU 19

**6.1 Install AWS CLI on Bastion and configure the aws configuration 19**

**6.2 Install kubectl 19**

**6.3 Install helm 19**

**6.4 Install docker 21**

**6.5 Accessing EKS 21**

**6.6 Docker tag and Push to ECR Registry 21**

**PART – 2**: **7**. **Containarization 21**

* 1. Dockerfiles For Backend Services 21
  2. Dockerfiles For Frontend Services 23

8. KUBERNETES YAMLS FOR BACKEND SERVICES 24

**8.1 configmaps 24**

**8.2 secrets 24**

**8.3 deployments 24**

**8.4 service 24**

**8.5 Helm charts for Backend service, Front-end Services and other applications of paywallet 24**

**9 Configure Ingress Controller 27**

* 1. **Configure dev-ingress.yaml 27**

**9.2 Configure uat-ingress.yaml 28**

**9.3 Configure prod-ingress.yaml 30**

**10. Jenkins Architecture 32**

**11. Jenkins Job configuration 33**

* 1. **Jenkins Build job for Front-end service 33**
  2. **Jenkins Build job for Back-end service 35**

**11.3 Jenkins deploy job for Back-end service 37**

**11.4 Jenkins deploy job for Back-end service 38**

**1: CREATING RESOURCES IN AWS**

* 1. Architecture Diagram Of Dev Environment

Diagram

Description automatically generated

* 1. Architecture Diagram Of Dev Environment

Diagram

Description automatically generated

* 1. Architecture Diagram Of Dev Environment

Diagram

Description automatically generated

2. CREATE RESOURCE GROUP

For Pre-Prod we are using “**N virginia**” region in Aws that holds related resources for pre-prod environment.

Region chosen for creating all resources: us-east-1

1. CREATE AURORA MYSQL SERVER AND AURORA POSTGRESQL DATABASES

**3.1. Create Aurora MySQL server for pre-prod environment with the name dev-fineract and uat-fineract with below specifications:**

**For Dev-Env: dev-fineract**Graphical user interface, text, application

Description automatically generated

Once Aurora MySQL server is created to connect to the database end points as follow:

|  |  |  |
| --- | --- | --- |
| **Endpoint name** | **Type** | **Port** |
| dev-fineract.cluster-ro-ckluizvytk2n.us-east-1.rds.amazonaws.com | Reader instance | 3306 |
| dev-fineract.cluster-ckluizvytk2n.us-east-1.rds.amazonaws.com | Writer instance | 3306 |

**For UAT-Env: uat-fineract**Graphical user interface, application

Description automatically generated

Once Aurora MySQL server is created to connect to the database end points as follow:

|  |  |  |
| --- | --- | --- |
| **Endpoint name** | **Type** | **Port** |
| uat-fineract.cluster-ckluizvytk2n.us-east-1.rds.amazonaws.com | Writer instance | 3306 |
| uat-fineract.cluster-ro-ckluizvytk2n.us-east-1.rds.amazonaws.com | Reader instance | 3306 |

**For Prod-Env: prod-fineract**

Graphical user interface, text, application

Description automatically generated

Once Aurora MySQL server is created to connect to the database end points as follow:

|  |  |  |
| --- | --- | --- |
| **Endpoint name** | **Type** | **Port** |
| prod-fineract.cluster-ckluizvytk2n.us-east-1.rds.amazonaws.com | Writer instance | 3306 |
| prod-fineract.cluster-ro-ckluizvytk2n.us-east-1.rds.amazonaws.com | Reader instance | 3306 |

**3.2 Create Aurora PostgreSQL server for pre-prod environment with the name dev-kong-konga and uat-kong-konga-keycloak with below specifications:**

**For Dev-Env: dev-kong-konga**

Graphical user interface, text, application

Description automatically generated

Once Aurora PostgreSQL server is created to connect to the database end points as follow:

|  |  |  |
| --- | --- | --- |
| **Endpoint name** | **Type** | **Port** |
| dev-kong-konga.cluster-ro-ckluizvytk2n.us-east-1.rds.amazonaws.com | Reader instance | 5432 |
| dev-kong-konga.cluster-ckluizvytk2n.us-east-1.rds.amazonaws.com | Writer instance | 5432 |

**For UAT-Env: uat-kong-konga-keycloak**

Graphical user interface, text, application

Description automatically generated

Once Aurora PostgreSQL server is created to connect to the database end points as follow:

|  |  |  |
| --- | --- | --- |
| **Endpoint name** | **Type** | **Port** |
| uat-kong-konga-keycloak.cluster-ro-ckluizvytk2n.us-east-1.rds.amazonaws.com | Reader instance | 5432 |
| uat-kong-konga-keycloak.cluster-ckluizvytk2n.us-east-1.rds.amazonaws.com | Writer instance | 5432 |

**For Prod-Env: prod-kong-konga-keycloak**

Graphical user interface, application

Description automatically generated

Once Aurora PostgreSQL server is created to connect to the database end points as follow:

|  |  |  |
| --- | --- | --- |
| **Endpoint name** | **Type** | **Port** |
| prod-kong-konga-keycloak.cluster-ro-ckluizvytk2n.us-east-1.rds.amazonaws.com | Reader instance | 5432 |
| prod-kong-konga-keycloak.cluster-ckluizvytk2n.us-east-1.rds.amazonaws.com | Writer instance | 5432 |

1. CREATE BASTION SERVER TO ACCESS TO CREATE AND ACCESS EACH ENVIRONMENT IN AWS

**For Dev-Bastion and UAT-Bastion**

|  |  |  |
| --- | --- | --- |
|  | **Dev Env** | **UAT Env** |
| **Platform details** | Ubuntu-20.04 | Ubuntu-20.04 |
| **Instance Type** | t2.medium(2 virtual CPUs for the instance, 4GB RAM) | t2.medium(2 virtual CPUs for the instance, 4GB RAM) |
| **VPC ID** | vpc-064d0f23d8d4a71f8(default VPC) | vpc-064d0f23d8d4a71f8(default VPC) |
| **Subnet ID** | subnet-0539a44eef0852cd9(us-east-1d) | subnet-0539a44eef0852cd9(us-east-1d) |
| **Public IP address(Instance restart IP changes accordingly)** | 34.229.79.165 | 54.205.33.235 |
| **Private IP addresses(Instance restart IP changes accordingly)** | 172.31.17.112 | 172.31.23.0 |
| **Root Volume Size** | 20GB | 20GB |
| **Additional Volume Disk** | No | No |
| **Key pairs** | root-bastion | root-bastion |
| **Security Group** | sg-093031c68703e0629(  default VPC security group) | sg-093031c68703e0629(  default VPC security group) |

**For Prod-Bastion**

|  |  |
| --- | --- |
|  | **Prod Env** |
| **Platform details** | Ubuntu-20.04 |
| **Instance Type** | t2.medium(2 virtual CPUs for the instance, 4GB RAM) |
| **VPC ID** | vpc-064d0f23d8d4a71f8 (default VPC) |
| **Subnet ID** | subnet-0539a44eef0852cd9(us-east-1d) |
| **Public IP address(Instance restart IP changes accordingly)** | 54.166.237.62 |
| **Private IP addresses(Instance restart IP changes accordingly)** | 172.31.20.208 |
| **Root Volume Size** | 20GB |
| **Additional Volume Disk** | No |
| **Key pairs** | root-bastion |
| **Security Group** | sg-093031c68703e0629(  default VPC security group) |

**For Jenkins-Server and SFTP-Dev**

|  |  |  |
| --- | --- | --- |
|  | **Jenkins server** | **SFTP-Dev** |
| **Platform details** | Ubuntu-20.04 | Ubuntu-20.04 |
| **Instance Type** | t2.medium(2 virtual CPUs for the instance, 4GB RAM) | t2.medium(2 virtual CPUs for the instance, 4GB RAM) |
| **VPC ID** | vpc-064d0f23d8d4a71f8(default VPC) | vpc-064d0f23d8d4a71f8(default VPC) |
| **Subnet ID** | subnet-0e096afdecd5aa4e5(us-east-1a) | subnet-0723dcb84ac110068(us-east-1c) |
| **Public IP address(Instance restart IP changes accordingly)** | 54.164.207.184 | 3.91.62.116 |
| **Private IP addresses(Instance restart IP changes accordingly)** | 172.31.42.195 | 172.31.83.251 |
| **Root Volume Size** | 50GB | 20GB |
| **Additional Volume Disk** | No | No |
| **Key pairs** | tf-jenkins-aws | root-bastion |
| **Security Group** | sg-053eca9617caaaf22(SG-Jenkins) | sg-093031c68703e0629(  default VPC security group) |

1. CREATE EKS CLUSTER AND AMAZON ELASTIC CONTAINER REGISTRY

**5.1 Install Terraform on Bastion.**

**Step 1- Register HashiCorp GPG keys**

$ curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -

**Step 2- Add HashiCorp package repository**

$ sudo apt-add-repository "deb [arch=$(dpkg --print-architecture)] https://apt.releases.hashicorp.com $(lsb\_release -cs) main"

**Step 3- Update "Ubuntu" packages list**

$ sudo apt update

**Step 4- Install Terraform on Ubuntu**

$ sudo apt install terraform

**Step 5 - Check Terraform version**

# Check version of Terraform

$ terraform -v

# Check PATH of Terraform

$ which terraform

**5.2 Install AWS CLI on Bastion and configure the aws configuration.**

$ sudo apt install awscli

Reference Document: <https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-linux.html>

$ Configure the Access key and Secret access key

* **export AWS\_ACCESS\_KEY\_ID=***<Generate the access key for each user>*
* **export AWS\_SECRET\_ACCESS\_KEY=***<Generate the secret access key for each user>*
* **export AWS\_DEFAULT\_REGION=***us-east-1*

**5.3 Terraform script to create EKS cluster from Ubuntu.**

GIT Hub Repo: <https://github.com/Maveric-Digital/devops-config/tree/master/Terraform-AWS>

Text

Description automatically generated

**eks-cluster.tf**

|  |
| --- |
| #  # EKS Cluster Resources  # \* IAM Role to allow EKS service to manage other AWS services  # \* EC2 Security Group to allow networking traffic with EKS cluster  # \* EKS Cluster  #  resource "aws\_iam\_role" "paywallet-cluster" {  name = "terraform-eks-paywallet-cluster"  assume\_role\_policy = <<POLICY  {  "Version": "2012-10-17",  "Statement": [  {  "Effect": "Allow",  "Principal": {  "Service": "eks.amazonaws.com"  },  "Action": "sts:AssumeRole"  }  ]  }  POLICY  }  resource "aws\_iam\_role\_policy\_attachment" "paywallet-cluster-AmazonEKSClusterPolicy" {  policy\_arn = "arn:aws:iam::aws:policy/AmazonEKSClusterPolicy"  role = aws\_iam\_role.paywallet-cluster.name  }  resource "aws\_iam\_role\_policy\_attachment" "paywallet-cluster-AmazonEKSVPCResourceController" {  policy\_arn = "arn:aws:iam::aws:policy/AmazonEKSVPCResourceController"  role = aws\_iam\_role.paywallet-cluster.name  }  resource "aws\_security\_group" "paywallet-cluster" {  name = "terraform-eks-paywallet-cluster"  description = "Cluster communication with worker nodes"  vpc\_id = aws\_vpc.paywallet.id  egress {  from\_port = 0  to\_port = 0  protocol = "-1"  cidr\_blocks = ["0.0.0.0/0"]  }  tags = {  Name = "terraform-eks-paywallet"  }  }  resource "aws\_security\_group\_rule" "paywallet-cluster-ingress-workstation-https" {  cidr\_blocks = [local.workstation-external-cidr]  description = "Allow workstation to communicate with the cluster API Server"  from\_port = 443  protocol = "tcp"  security\_group\_id = aws\_security\_group.paywallet-cluster.id  to\_port = 443  type = "ingress"  }  resource "aws\_eks\_cluster" "paywallet" {  name = var.cluster-name  role\_arn = aws\_iam\_role.paywallet-cluster.arn  vpc\_config {  security\_group\_ids = [aws\_security\_group.paywallet-cluster.id]  subnet\_ids = aws\_subnet.paywallet[\*].id  }  depends\_on = [  aws\_iam\_role\_policy\_attachment.paywallet-cluster-AmazonEKSClusterPolicy,  aws\_iam\_role\_policy\_attachment.paywallet-cluster-AmazonEKSVPCResourceController,  ]  } |

**eks-worker-nodes.tf**

|  |
| --- |
| #  # EKS Worker Nodes Resources  # \* IAM role allowing Kubernetes actions to access other AWS services  # \* EKS Node Group to launch worker nodes  #  resource "aws\_iam\_role" "paywallet-node" {  name = "terraform-eks-paywallet-node"  assume\_role\_policy = <<POLICY  {  "Version": "2012-10-17",  "Statement": [  {  "Effect": "Allow",  "Principal": {  "Service": "ec2.amazonaws.com"  },  "Action": "sts:AssumeRole"  }  ]  }  POLICY  }  resource "aws\_iam\_role\_policy\_attachment" "paywallet-node-AmazonEKSWorkerNodePolicy" {  policy\_arn = "arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"  role = aws\_iam\_role.paywallet-node.name  }  resource "aws\_iam\_role\_policy\_attachment" "paywallet-node-AmazonEKS\_CNI\_Policy" {  policy\_arn = "arn:aws:iam::aws:policy/AmazonEKS\_CNI\_Policy"  role = aws\_iam\_role.paywallet-node.name  }  resource "aws\_iam\_role\_policy\_attachment" "paywallet-node-AmazonEC2ContainerRegistryReadOnly" {  policy\_arn = "arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly"  role = aws\_iam\_role.paywallet-node.name  }  resource "aws\_eks\_node\_group" "paywallet" {  cluster\_name = aws\_eks\_cluster.paywallet.name  node\_group\_name = "paywallet"  node\_role\_arn = aws\_iam\_role.paywallet-node.arn  subnet\_ids = aws\_subnet.paywallet[\*].id  scaling\_config {  desired\_size = 1  max\_size = 2  min\_size = 1  }  depends\_on = [  aws\_iam\_role\_policy\_attachment.paywallet-node-AmazonEKSWorkerNodePolicy,  aws\_iam\_role\_policy\_attachment.paywallet-node-AmazonEKS\_CNI\_Policy,  aws\_iam\_role\_policy\_attachment.paywallet-node-AmazonEC2ContainerRegistryReadOnly,  ]  } |

**providers.tf**

|  |
| --- |
| terraform {  required\_version = ">= 0.12"  }  provider "aws" {  region = var.aws\_region  }  data "aws\_availability\_zones" "available" {}  # Not required: currently used in conjunction with using  # icanhazip.com to determine local workstation external IP  # to open EC2 Security Group access to the Kubernetes cluster.  # See workstation-external-ip.tf for additional information.  provider "http" {} |

**variables.tf**

|  |
| --- |
| variable "aws\_region" {  default = "us-east-1"  }  variable "cluster-name" {  default = "eks-paywallet"  type = string  } |

**vpc.tf**

|  |
| --- |
| #  # VPC Resources  # \* VPC  # \* Subnets  # \* Internet Gateway  # \* Route Table  #  resource "aws\_vpc" "paywallet" {  cidr\_block = "10.0.0.0/16"  tags = tomap({  "Name" = "terraform-eks-paywallet-node",  "kubernetes.io/cluster/${var.cluster-name}" = "shared",  })  }  resource "aws\_subnet" "paywallet" {  count = 2  availability\_zone = data.aws\_availability\_zones.available.names[count.index]  cidr\_block = "10.0.${count.index}.0/24"  map\_public\_ip\_on\_launch = true  vpc\_id = aws\_vpc.paywallet.id  tags = tomap({  "Name" = "terraform-eks-paywallet-node",  "kubernetes.io/cluster/${var.cluster-name}" = "shared",  })  }  resource "aws\_internet\_gateway" "paywallet" {  vpc\_id = aws\_vpc.paywallet.id  tags = {  Name = "terraform-eks-paywallet"  }  }  resource "aws\_route\_table" "paywallet" {  vpc\_id = aws\_vpc.paywallet.id  route {  cidr\_block = "0.0.0.0/0"  gateway\_id = aws\_internet\_gateway.paywallet.id  }  }  resource "aws\_route\_table\_association" "paywallet" {  count = 2  subnet\_id = aws\_subnet.paywallet.\*.id[count.index]  route\_table\_id = aws\_route\_table.paywallet.id  } |

**workstation-external-ip.tf**

|  |
| --- |
| #  # Workstation External IP  #  # This configuration is not required and is  # only provided as an example to easily fetch  # the external IP of your local workstation to  # configure inbound EC2 Security Group access  # to the Kubernetes cluster.  #  data "http" "workstation-external-ip" {  url = "http://ipv4.icanhazip.com"  }  # Override with variable or hardcoded value if necessary  locals {  workstation-external-cidr = "${chomp(data.http.workstation-external-ip.body)}/32"  } |

**outputs.tf**

|  |
| --- |
| #  # Outputs  #  locals {  config\_map\_aws\_auth = <<CONFIGMAPAWSAUTH  apiVersion: v1  kind: ConfigMap  metadata:  name: aws-auth  namespace: kube-system  data:  mapRoles: |  - rolearn: ${aws\_iam\_role.paywallet-node.arn}  username: system:node:{{EC2PrivateDNSName}}  groups:  - system:bootstrappers  - system:nodes  CONFIGMAPAWSAUTH  kubeconfig = <<KUBECONFIG  apiVersion: v1  clusters:  - cluster:  server: ${aws\_eks\_cluster.paywallet.endpoint}  certificate-authority-data: ${aws\_eks\_cluster.paywallet.certificate\_authority[0].data}  name: kubernetes  contexts:  - context:  cluster: kubernetes  user: aws  name: aws  current-context: aws  kind: Config  preferences: {}  users:  - name: aws  user:  exec:  apiVersion: client.authentication.k8s.io/v1alpha1  command: aws-iam-authenticator  args:  - "token"  - "-i"  - "${var.cluster-name}"  KUBECONFIG  }  output "config\_map\_aws\_auth" {  value = local.config\_map\_aws\_auth  }  output "kubeconfig" {  value = local.kubeconfig  } |

**Commands to run the terraform script**

* $ terraform init
* $ terraform validate
* $ terraform plan
* $ terraform apply -auto-approve

**Note: Once Create take the statefile and upload to s3 bucket**

* 1. **Create the ECR Registry.**

<https://console.aws.amazon.com/ecr/repositories?region=us-east-1>

Create a Private Repository for each service

Graphical user interface, text, application, email

Description automatically generated

6 ACCESSING EKS CLUSTER FROM UBUNTU

**6.1 Install AWS CLI on Bastion and configure the aws configuration.**

$ sudo apt install awscli

Reference Document: <https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-linux.html>

$ Configure the Access key and Secret access key

* **export AWS\_ACCESS\_KEY\_ID=***<Generate the access key for each user>*
* **export AWS\_SECRET\_ACCESS\_KEY=***<Generate the secret access key for each user>*
* **export AWS\_DEFAULT\_REGION=***us-east-1*

**6.2 Install kubectl**

$ curl -LO [https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl](https://dl.k8s.io/release/$(curl%20-L%20-s%20https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl)

$ curl -LO [https://dl.k8s.io/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256](https://dl.k8s.io/$(curl%20-L%20-s%20https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256)

$ echo "$(<kubectl.sha256) kubectl" | sha256sum --check

kubectl: OK

$ sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

$ chmod +x kubectl

$ mkdir -p ~/.local/bin/kubectl

$ mv ./kubectl ~/.local/bin/kubectl

$ kubectl version --client

**6.3 Install helm**

$ curl https://baltocdn.com/helm/signing.asc | sudo apt-key add –

$ sudo apt-get install apt-transport-https –yes

$ echo "deb https://baltocdn.com/helm/stable/debian/ all main" | sudo tee /etc/apt/sources.list.d/helm-stable-debian.list

$ sudo apt-get update

$ sudo apt-get install helm

* 1. **Install docker**

$ sudo apt-get update

$ sudo apt-get install ca-certificates curl gnupg lsb-release

$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

$ sudo apt-get update

$ sudo apt-get install docker-ce docker-ce-cli containerd.io

* 1. **Accessing EKS:**

**For Dev-Env:**

* aws eks --region us-east-1 update-kubeconfig --name eks-paywallet

**For UAT-Env:**

* aws eks --region us-east-1 update-kubeconfig --name eks-paywallet-uat

**For Prod-Env:**

* aws eks --region us-east-1 update-kubeconfig --name eks-paywallet-prod

**6.6 Docker tag and Push to ECR Registry**

* docker tag <service-name>:latest 669171879954.dkr.ecr.us-east-1.amazonaws.com/<service-name>:<GIT\_latest\_tag>
* docker push 669171879954.dkr.ecr.us-east-1.amazonaws.com/<service-name>:<GIT\_latest\_tag>

7 CONTAINERIZATION

* 1. Dockerfiles For Backend Services

To run the micro-services in containers with Docker, first clone the repository from Github and build it (mvn clean install -DskipTests). Now write a Dockerfile to copy the resultant war file into container and run it using java –jar command. Place the Dockerfile under the micro-services repo of your project.

**pw-account-service:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/account-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9050  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**pw-account-transaction-service:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/account-transaction-1.0-SNAPSHOT.jar app.jar  EXPOSE 9100  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**pw-ach-processing-service:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/ach-processing-0.0.1-SNAPSHOT.jar app.jar  RUN mkdir -p /ach/incoming/ /ach/outgoing/  EXPOSE 8080  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**pw-ach-validation-service:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/ach-validation-0.0.1-SNAPSHOT.jar app.jar  RUN mkdir -p /ach/incoming/ /ach/outgoing/  EXPOSE 9120  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**pw-borrower-service:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/borrower-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 9060  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**pw-dashboard-service:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/dashboard-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 8080  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**pw-employer-service:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/employer-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 8080  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**pw-event-logs-service:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/event-logs-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 8080  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**pw-link-service:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/ link-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 8080  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**pw-mock-service:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/mock-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 8080  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**pw-notification-service:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/notification-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 8080  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**pw-otp-service:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/otp-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 8080  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**pw-pay-allocation-service:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/pay-allocation-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 8080  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**pw-request-service:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/request-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 8080  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

**pw-underwriting-service:**

|  |
| --- |
| FROM openjdk:11-jre-slim  COPY target/underwriting-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 8080  CMD ["java","-Dspring.profiles.active=dev","-jar","app.jar"] |

7.2 Dockerfiles For Frontend Services

As the frontend is ReactJS application, it requires npm build in order to start the application. Build the npm package on the on-prem server and copy the resultant build to docker image.

**atomicfi-micro-app:**

|  |
| --- |
| FROM node:12.2.0-alpine  RUN mkdir /app  WORKDIR /app  COPY package.json ./  RUN npm install  COPY . /app  RUN npm run build  EXPOSE 3001  CMD ["npm", "start"] |

**argyle-micro-app:**

|  |
| --- |
| FROM node:12.2.0-alpine  RUN mkdir /app  WORKDIR /app  COPY package.json ./  RUN npm install  COPY . /app  RUN npm run build  EXPOSE 3002  CMD ["npm", "start"] |

**borrower-verification-micro-app:**

|  |
| --- |
| FROM node:12.2.0-alpine  RUN mkdir /app  WORKDIR /app  COPY package.json ./  RUN npm install  COPY . /app  RUN npm run build  EXPOSE 3000  CMD ["npm", "start"] |

**customer-account-verification-micro-app:**

|  |
| --- |
| FROM node:12.2.0-alpine  RUN mkdir /app  WORKDIR /app  COPY package.json ./  RUN npm install  COPY . /app  RUN npm run build  EXPOSE 3003  CMD ["npm", "start"] |

**paywallet-developer-portal:**

|  |
| --- |
| FROM node:12.2.0-alpine  RUN mkdir /app  WORKDIR /app  COPY package.json ./  RUN npm install  COPY . /app  RUN npm run build  EXPOSE 3050  CMD ["npm", "start"] |

8 KUBERNETES YAMLS FOR BACKEND SERVICES

Below are the K8s resources being used for Paywallet:

1. Configmap
2. Secrets
3. Deployment
4. Service
   1. **Configmap:**

Create a configmap to externalize the properties present in application-<env>.properties and application yaml file.

How to add key values in configmap for spring boot application?

For example, to store DB URL in configmap, instead of simply specifying url, you should specify fully qualified path to key like “multitenancy.mtapp.master.datasource.url” and provide its value separated by a colon. Likewise specify all key-value pairs under data section inside configmap.

* 1. **Secrets:**

In secrets, we are storing DB password in encoded format. As DB password remains same for all backend services, we create a common secret

Encode the Mysql server password using “ echo -n "<>" | base64 ”, we will get an encoded string which will be used inside secrets.yaml.

* 1. **Deployments:**

As we will push images to Elastic container registry (ECR), properly tag the image according to the login server name of your ECR instance.

How to tag an image to push to ECR?

Below is the image tagging format to publish an image to ECR

<aws\_account\_id>.dkr.ecr.us-east-1.amazonaws.com/<image\_name>:<tag>

* 1. **Services:**

A Service in Kubernetes is a REST object, similar to a Pod. Like all of the REST objects, you can POST a Service definition to the API server to create a new instance.

* 1. **Helm charts for Backend service, Front-end Services and other applications of paywallet.**
* Helm chart for pw-account-service: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/account>
* Helm chart for pw-account-transaction-service: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/account-transaction>
* Helm chart for pw-ach-processing-service: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/ach-processing>
* Helm chart for pw-ach-validation-service: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/ach-validation>
* Helm chart for pw-borrower-service: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/borrower>
* Helm chart for pw-dashboard-service: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/dashboard>
* Helm chart for pw-employer-service: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/employer>
* Helm chart for pw-event-logs-service: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/event-logs>
* Helm chart for pw-link-service: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/link>
* Helm chart for pw-mock-service: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/mock>
* Helm chart for pw-notification-service: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/notification>
* Helm chart for pw-otp-service: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/otp>
* Helm chart for pw-pay-allocation-service: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/pay-allocation>
* Helm chart for pw-request-service: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/request>
* Helm chart for pw-underwriting-service: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/underwriting>
* Helm chart for atomicfi-micro-app: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/atomicfi-micro-app>
* Helm chart for argyle-micro-app: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/argyle-micro-app>
* Helm chart for borrower-verification-micro-app: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/borrower-verification-micro-app>
* Helm chart for customer-account-verification-micro-app: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/customer-account-verification-micro-app>
* Helm chart for paywallet-developer-portal: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/paywallet-developer-portal>
* Helm chart for Fineract: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/fineract>
* Helm chart for Kong: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/kong>
* Helm chart for Konga: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/konga>
* Helm chart for Keycloak: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/keycloak>
* Helm chart for MongoDb: <https://github.com/Maveric-Digital/devops-config/tree/master/helm/mongodb>

1. CONFIGURE INGRESS CONTROLLER

**9.1 Configure dev-ingress.yaml:**

|  |
| --- |
| apiVersion: extensions/v1beta1  kind: Ingress  metadata:  name: payuw-ingress  annotations:  kubernetes.io/ingress.class: nginx  alb.ingress.kubernetes.io/scheme: internet-facing  alb.ingress.kubernetes.io/target-type: instance  alb.ingress.kubernetes.io/security-groups: sg-09960e517541935ab  nginx.ingress.kubernetes.io/rewrite-target: /$2  spec:  tls:  - hosts:  - sandbox2-dev.paywalletllc.com  secretName: sandbox2-dev  rules:  - host: sandbox2-dev.paywalletllc.com  http:  paths:  - path: /eureka(/|$)(.\*)  backend:  serviceName: eureka-server  servicePort: 8761  - path: /(|$)(.\*)  backend:  serviceName: kong-kong-proxy  servicePort: 80  - path: /konga(/|$)(.\*)  backend:  serviceName: konga  servicePort: 80  - path: /keycloak(/|$)(.\*)  backend:  serviceName: keycloak2-http  servicePort: 80  - path: /fineract(/|$)(.\*)  backend:  serviceName: fineract  servicePort: 80  - path: /fineract-provider(/|$)(.\*)  backend:  serviceName: fineract  servicePort: 80 |

* 1. **Configure uat-ingress.yaml:**

|  |
| --- |
| apiVersion: extensions/v1beta1  kind: Ingress  metadata:  name: payuw-uat-ingress  annotations:  kubernetes.io/ingress.class: nginx  alb.ingress.kubernetes.io/scheme: internet-facing  alb.ingress.kubernetes.io/target-type: instance  alb.ingress.kubernetes.io/security-groups: sg-02c6ab9a941b1d2eb  nginx.ingress.kubernetes.io/rewrite-target: /$2  spec:  tls:  - hosts: ["keycloak-uat.paywalletllc.com", "konga-uat.paywalletllc.com", "fineract-uat.paywalletllc.com", "services-uat.paywalletllc.com", "bvr-uat.paywalletllc.com", "provider1-uat.paywalletllc.com", "provider2-uat.paywalletllc.com", "custactver-uat.paywalletllc.com"]  secretName: sandbox2-uat  rules:  - host: uat-portal.paywalletllc.com  http:  paths:  - path: /(|$)(.\*)  backend:  serviceName: paywallet-developer-portal  servicePort: 80  - host: keycloak-uat.paywalletllc.com  http:  paths:  - path: /(|$)(.\*)  backend:  serviceName: keycloak-http  servicePort: 80  - host: konga-uat.paywalletllc.com  http:  paths:  - path: /(|$)(.\*)  backend:  serviceName: konga  servicePort: 80  - host: fineract-uat.paywalletllc.com  http:  paths:  - path: /fineract/\*  backend:  serviceName: fineract  servicePort: 80  - host: services-uat.paywalletllc.com  http:  paths:  - path: /(|$)(.\*)  backend:  serviceName: kong-kong-proxy  servicePort: 80  - host: bvr-uat.paywalletllc.com  http:  paths:  - path: /(|$)(.\*)  backend:  serviceName: borrower-verification-micro-app  servicePort: 80  - host: provider1-uat.paywalletllc.com  http:  paths:  - path: /(|$)(.\*)  backend:  serviceName: atomicfi-micro-app  servicePort: 80  - host: provider2-uat.paywalletllc.com  http:  paths:  - path: /(|$)(.\*)  backend:  serviceName: argyle-micro-app  servicePort: 80  - host: custactver-uat.paywalletllc.com  http:  paths:  - path: /(|$)(.\*)  backend:  serviceName: customer-account-verification-micro-app  servicePort: 80 |

* 1. **Configure prod-ingress.yaml:**

|  |
| --- |
| apiVersion: extensions/v1beta1  kind: Ingress  metadata:  name: payuw-prod-ingress  annotations:  kubernetes.io/ingress.class: nginx  alb.ingress.kubernetes.io/scheme: internet-facing  alb.ingress.kubernetes.io/target-type: instance  alb.ingress.kubernetes.io/security-groups: sg-072ba05cc66e7832e  nginx.ingress.kubernetes.io/rewrite-target: /$2  spec:  tls:  - hosts: ["idp.paywalletllc.com", "gateway.paywalletllc.com"]  secretName: prod-tls-secret  rules:  - host: idp.paywalletllc.com  http:  paths:  - path: /(|$)(.\*)  backend:  serviceName: keycloak-http  servicePort: 80  - host: gateway.paywalletllc.com  http:  paths:  - path: /(|$)(.\*)  backend:  serviceName: konga  servicePort: 80  - host: services.paywalletllc.com  http:  paths:  - path: /(|$)(.\*)  backend:  serviceName: kong-kong-proxy  servicePort: 80  - host: prod-portal.paywalletllc.com  http:  paths:  - path: /(|$)(.\*)  backend:  serviceName: paywallet-developer-portal  servicePort: 80  - host: va.paywalletllc.com  http:  paths:  - path: /\*  backend:  serviceName: fineract  servicePort: 80  - host: payments.payalletllc.com  http:  paths:  - path: /borrower(|$)(.\*)  backend:  serviceName: borrower-verification-micro-app  servicePort: 80  - host: payments.payalletllc.com  http:  paths:  - path: /provider1(|$)(.\*)  backend:  serviceName: atomicfi-micro-app  servicePort: 80  - host: payments.payalletllc.com  http:  paths:  - path: /provider2(|$)(.\*)  backend:  serviceName: argyle-micro-app  servicePort: 80  - host: payments.payalletllc.com  http:  paths:  - path: /allocation(|$)(.\*)  backend:  serviceName: customer-account-verification-micro-app  servicePort: 80 |

1. JENKINS ARCHITECTURE

Graphical user interface

Description automatically generated with low confidence

The Build for Dev, UAT and Prod environments is configured in such a way that we will pull the code from micro service repo, maven build/ npm build , build docker image, push the resultant docker image to ECR.

**For each environment**, we are have separate deployment folder created in jenkins. we will deploy specific version of docker image to the respective environment. So, here, we will pull specific version of docker image from ECR repo and deploy it to respective EKS cluster.

The Build for Dev, UAT and Prod environments is configured in such a way that we will pull the code from micro service repo, maven build/ npm build , build docker image, push the resultant docker image to ECR.

Jenkins URL: <https://jenkins.paywalletllc.com/>

1. JENKINS JOB CONFIGURATION

* Here we are using <https://jenkins.paywalletllc.com/> server (with label name: Jenkins server in aws ec2)
* First, configure Elastic container registry (ECR) credentials and Github repo credentials in jenkins global credentials with kind Username and password

Create a Build jenkins job of type “pipeline”.

**11.1 Jenkins Build job for Front-end service:**

|  |
| --- |
| pipeline {  agent any  environment {  BUILD\_TRIGGER\_BY = "${currentBuild.getBuildCauses()[0].shortDescription}"  AWS\_ACCOUNT\_ID="669171879954"  AWS\_DEFAULT\_REGION="us-east-1"  GIT\_COMMIT\_REV=''  REPOSITORY\_URI = "${AWS\_ACCOUNT\_ID}.dkr.ecr.${AWS\_DEFAULT\_REGION}.amazonaws.com"  }  parameters {  string(name: 'Branch', defaultValue: 'development', description: 'Enter the branch to deploy?')  choice(name: 'Service', choices: ['atomicfi-micro-app', 'argyle-micro-app', 'borrower-verification-micro-app', 'customer-account-verification-micro-app', 'paywallet-developer-portal'], description: 'Select the service you want to deploy')  choice(name: 'environment', choices: ['dev', 'uat'], description: 'Select the environment you want to deploy')  }  stages {  stage('Logging into AWS ECR') {  steps {  script {  sh "aws ecr get-login-password --region ${AWS\_DEFAULT\_REGION} | docker login --username AWS --password-stdin ${AWS\_ACCOUNT\_ID}.dkr.ecr.${AWS\_DEFAULT\_REGION}.amazonaws.com"  }  }  }  stage('Cloning our Git') {  steps {  git branch: '$Branch', credentialsId: 'Github-Key', url: 'https://github.com/Maveric-Digital/${Service}.git'  }  }    stage('Configure') {  steps {  script {  GIT\_COMMIT\_REV = sh(returnStdout: true, script: "git log -n 1 --pretty=format:'%h'").trim()  }  }  }    // Building Docker images  stage('Building image') {  steps{  script {  dockerImage = docker.build "${Service}:${GIT\_COMMIT\_REV}"  }  }  }  // Uploading Docker images into AWS ECR  stage('Pushing to ECR') {  steps{  script {  sh "docker tag ${Service}:$GIT\_COMMIT\_REV ${REPOSITORY\_URI}/${Service}:$GIT\_COMMIT\_REV"  sh "docker push ${AWS\_ACCOUNT\_ID}.dkr.ecr.${AWS\_DEFAULT\_REGION}.amazonaws.com/${Service}:$GIT\_COMMIT\_REV"  sh "docker rmi ${Service}:$GIT\_COMMIT\_REV ${AWS\_ACCOUNT\_ID}.dkr.ecr.${AWS\_DEFAULT\_REGION}.amazonaws.com/${Service}:$GIT\_COMMIT\_REV"  }  }  }  }  post {  success {  slackSend (color: '#00FF00', message: "Successfully pushed the ${Service} dokcer image for ${environment} environment, ${BUILD\_TRIGGER\_BY}. Docker image version for this build is $GIT\_COMMIT\_REV' (${env.BUILD\_URL})")  }  failure {  slackSend (color: '#FF0000', message: "Failed to pushed the ${Service} dokcer image for ${environment} environment, ${BUILD\_TRIGGER\_BY}. Docker image version for this build is $GIT\_COMMIT\_REV' (${env.BUILD\_URL})")  }  always {  cleanWs()  }  }  } |

**Parameter name - Default value**

Branch - Enter the branch to build docker image

Service - From choice select the micro-service to build

Environment - Select the respective env to build

**11.2** **Jenkins Build job for Back-end service:**

|  |
| --- |
| pipeline {  agent any  environment {  BUILD\_TRIGGER\_BY = "${currentBuild.getBuildCauses()[0].shortDescription}"  AWS\_ACCOUNT\_ID="669171879954"  AWS\_DEFAULT\_REGION="us-east-1"  GIT\_COMMIT\_REV=''  envr=''  REPOSITORY\_URI = "${AWS\_ACCOUNT\_ID}.dkr.ecr.${AWS\_DEFAULT\_REGION}.amazonaws.com"  }  parameters {  string(name: 'Branch', defaultValue: 'develop', description: 'Enter the branch to deploy?')  choice(name: 'Service', choices: ['ach-file-handling-service', 'ach-processing-service', 'helper-service', 'pw-account-service', 'pw-account-transaction-service', 'pw-ach-processing-service', 'pw-ach-validation-service', 'pw-borrower-service', 'pw-dashboard-service', 'pw-employer-service', 'pw-event-logs-service', 'pw-link-service', 'pw-mock-service', 'pw-notification-service', 'pw-otp-service', 'pw-pay-allocation-service', 'pw-provider-service', 'pw-request-service', 'pw-underwriting-service'], description: 'Select the service you want to deploy')  choice(name: 'environment', choices: ['dev', 'uat', 'prod'], description: 'Select the environment you want to deploy')  }  stages {  stage('Logging into AWS ECR') {  steps {  script {  sh "aws ecr get-login-password --region ${AWS\_DEFAULT\_REGION} | docker login --username AWS --password-stdin ${AWS\_ACCOUNT\_ID}.dkr.ecr.${AWS\_DEFAULT\_REGION}.amazonaws.com"  }  }  }  stage('Cloning our Git') {  steps {  git branch: '$Branch', credentialsId: 'Github-Key', url: 'https://github.com/Maveric-Digital/${Service}.git'  sh'''  sed -i "s/dev/$environment/g" Dockerfile  '''  }  }    stage('Configure') {  steps {  script {  GIT\_COMMIT\_REV = sh(returnStdout: true, script: "git log -n 1 --pretty=format:'%h'").trim()  }  }  }    stage ('Maven Build') {  steps {  sh """  mvn clean install -DskipTests  """  }  }    // Building Docker images  stage('Building image') {  steps{  script {  dockerImage = docker.build "${Service}:${GIT\_COMMIT\_REV}"  }  }  }  // Uploading Docker images into AWS ECR  stage('Pushing to ECR') {  steps{  script {  sh "docker tag ${Service}:$GIT\_COMMIT\_REV ${REPOSITORY\_URI}/${Service}:$GIT\_COMMIT\_REV"  sh "docker push ${AWS\_ACCOUNT\_ID}.dkr.ecr.${AWS\_DEFAULT\_REGION}.amazonaws.com/${Service}:$GIT\_COMMIT\_REV"  sh "docker rmi ${Service}:$GIT\_COMMIT\_REV ${AWS\_ACCOUNT\_ID}.dkr.ecr.${AWS\_DEFAULT\_REGION}.amazonaws.com/${Service}:$GIT\_COMMIT\_REV"  }  }  }  }  post {  success {  slackSend (color: '#00FF00', message: "Successfully pushed the ${Service} docker image for ${environment} environment, ${BUILD\_TRIGGER\_BY}. Docker image version for this build is $GIT\_COMMIT\_REV' (${env.BUILD\_URL})")  }  failure {  slackSend (color: '#FF0000', message: "Failed to pushed the ${Service} docker image for ${environment} environment, ${BUILD\_TRIGGER\_BY}. Docker image version for this build is $GIT\_COMMIT\_REV' (${env.BUILD\_URL})")  }  always {  cleanWs()  }  }  } |

**Parameter name - Default value**

Branch - Enter the branch to build docker image

Service - From choice select the micro-service to build

Environment - Select the respective env to build

* 1. **Jenkins deploy job for Back-end service:**

|  |
| --- |
| pipeline {  agent any  environment {  BUILD\_TRIGGER\_BY = "${currentBuild.getBuildCauses()[0].shortDescription}"  }  parameters {  choice(name: 'Service', choices: ['account', 'account-transaction', 'ach-file-handling', 'ach-processing', 'ach-validation', 'borrower', 'dashboard', 'employer', 'event-logs', 'helper', 'link', 'mock', 'notification', 'otp', 'pay-allocation', 'provider', 'request', 'underwriting'], description: 'Select the service you want to deploy')  string(name: 'environment', defaultValue: 'dev', description: 'Select the environment you want to deploy')  string(name: 'Docker\_tag', description: 'Enter the Docker image version to deploy')  }  stages {  stage('Setup Dev cluster') {  steps {  sh '''  aws eks --region us-east-1 update-kubeconfig --name eks-paywallet  '''  }  }  stage('Cloning our helm chart for the service') {  steps {  git credentialsId: 'Github-Key', url: 'https://github.com/Maveric-Digital/devops-config.git'  }  }  stage('Deploy the helm chart for the service ') {  steps{  script {  sh "helm upgrade --install ${Service} helm/${Service} -n ${environment} -f helm/${Service}/${environment}.yaml --set imageTag=${Docker\_tag}"  }  }  }  }  post {  success {  slackSend (color: '#008000', message: "Deployed ${Service}-service in ${environment} environment ${BUILD\_TRIGGER\_BY} (${env.BUILD\_URL})")  }  failure {  slackSend (color: '#FF0000', message: "Deployment failed for ${Service}-service in ${environment} environment ${BUILD\_TRIGGER\_BY} (${env.BUILD\_URL})")  }  always {  cleanWs()  }  }  } |

**Parameter name - Default value**

Service - From choice select the micro-service to build

environment - Select the respective env to build

Docker\_tag - Enter the Docker image version to deploy

* 1. **Jenkins deploy job for Back-end service:**

|  |
| --- |
| pipeline {  agent any  environment {  BUILD\_TRIGGER\_BY = "${currentBuild.getBuildCauses()[0].shortDescription}"  }  parameters {  choice(name: 'Service', choices: ['atomicfi-micro-app', 'argyle-micro-app', 'borrower-verification-micro-app', 'customer-account-verification-micro-app', 'paywallet-developer-portal'], description: 'Select the service you want to deploy')  string(name: 'environment', defaultValue: 'dev', description: 'Select the environment you want to deploy')  string(name: 'Docker\_tag', description: 'Enter the Docker image version to deploy')  }  stages {  stage('Setup Dev cluster') {  steps {  sh '''  aws eks --region us-east-1 update-kubeconfig --name eks-paywallet  '''  }  }  stage('Cloning our helm chart for the service') {  steps {  git credentialsId: 'Github-Key', url: 'https://github.com/Maveric-Digital/devops-config.git'  }  }  stage('Deploy the helm chart for the service ') {  steps{  script {  sh "helm upgrade --install ${Service} helm/${Service} --values helm/${Service}/${environment}.yaml --set imageTag=${Docker\_tag} -n ${environment}"  }  }  }  }  post {  success {  slackSend (color: '#008000', message: "Deployed ${Service}-service in ${environment} environment ${BUILD\_TRIGGER\_BY} (${env.BUILD\_URL})")  }  failure {  slackSend (color: '#FF0000', message: "Deployment failed for ${Service}-service in ${environment} environment ${BUILD\_TRIGGER\_BY} (${env.BUILD\_URL})")  }  always {  cleanWs()  }  }  } |

**Parameter name - Default value**

Service - From choice select the micro-service to build

environment - Select the respective env to build

Docker\_tag - Enter the Docker image version to deploy

**To Deploy on UAT and Prod, change the cluster name in Setup Dev cluster stage in the above pipeline script.**

**For UAT:**

|  |
| --- |
| stages {  stage('Setup uat cluster') {  steps {  sh '''  aws eks --region us-east-1 update-kubeconfig --name eks-paywallet-uat  '''  }  } |

**For Prod:**

|  |
| --- |
| stages {  stage('Setup Prod cluster') {  steps {  sh '''  aws eks --region us-east-1 update-kubeconfig --name eks-paywallet-prod  '''  }  } |