CANDIDATE NAME: **HUYNH THAI BAO**

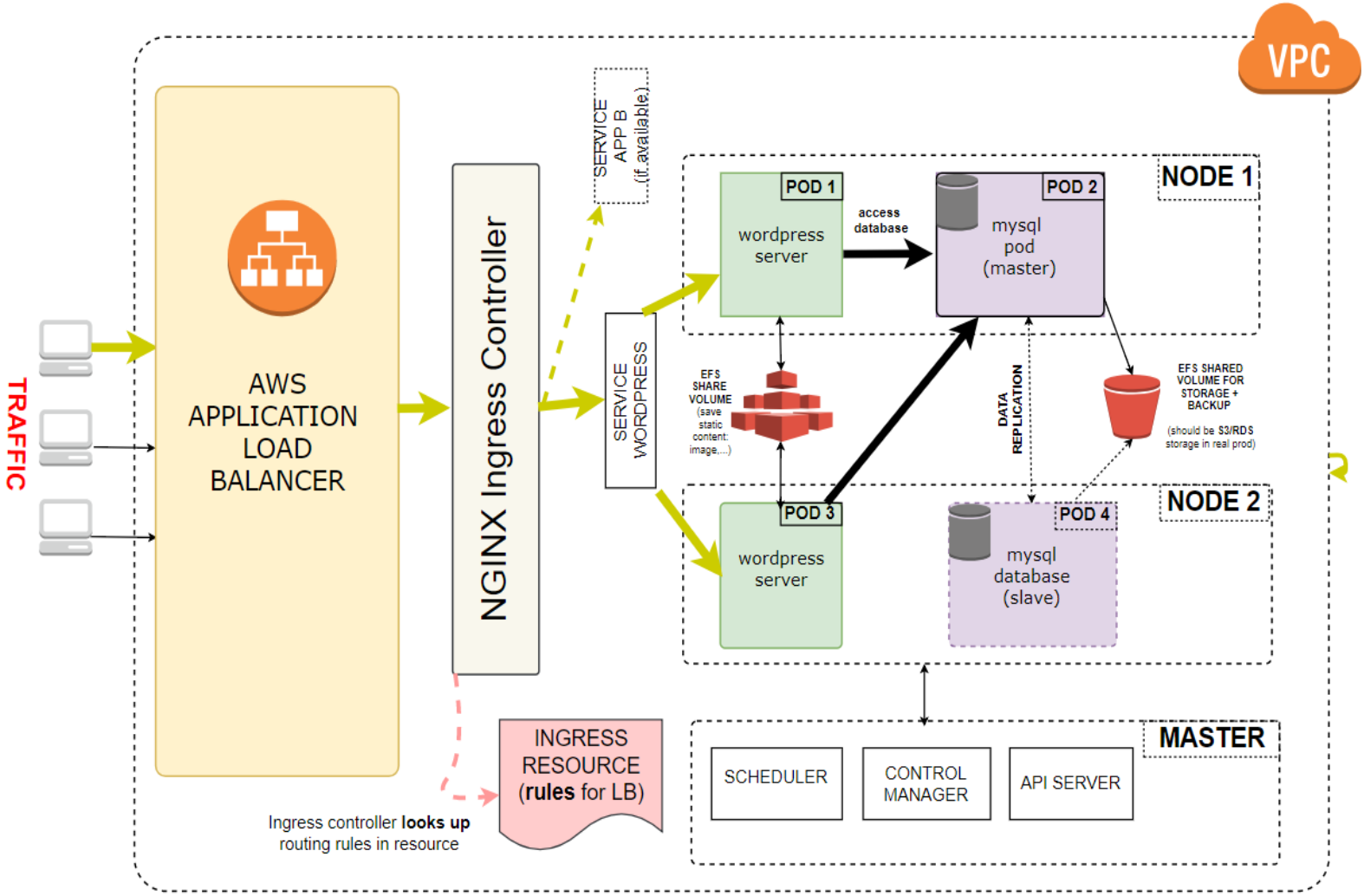
APPLICANT POSITION: **SYSTEM ENGINEER (CHOTOT Co.Ltd)**

SUBMIT DAY: **28/08/2018**

REQUIREMENTS & ASSIGNMENT COMPLETION RATE.

No.	Requirement	Completion	Note
1	Setup Kubernetes on AWS EC2 (1 master + 2 nodes)	✓	
2	Setup etcd cluster	✓	
3	Deploy Wordpress on Kubernetes cluster with default page /	✓	
4	Use Ingress for Application Load Balancing: + Deploy ingress controller + Deploy ingress resource for path-based load balancing	✓	
5	+ Using /etc/hosts to manually add alias domain name (chotot.kubes.com) for aws-public-ip + Access wordpress successfully via that hostname	✓	+ Done by local testing + Waiting for your turn ☺

FULL ARCHITECTURE

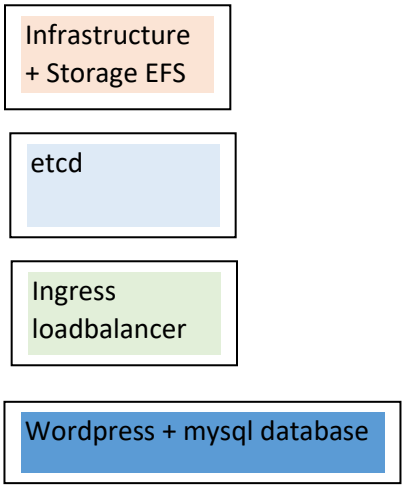


SYSTEM OVERVIEW:

1) 1 Master node (for control, schedule, sending API to workers,..)
2) 2 worker nodes for executing the replicas tasks to ensure the High-availability system (in this case: running database mysql & frontend wordpress webserver).
3) Deploy etcd as key-value store application for the whole cluster.
4) EFS is used as shared storage between pods (wordpress, mysql) in kubernetes cluster
   In the real production: only EFS is used for storage static data (images, thumbnails,…) and S3/RDS/DynamoDB are suitable for database.
   However, for simplicity, I used EFS for both wordpress + mysql in my case.
5) Ingress controller to manipulate inputs from ingress resource, which specifies rules for Application Load Balancer (path-based load banlacing,…).

IMPLEMENTATION:

```
.
+--- efs_provision
|   +--- efs-provisioner-mysql.yaml
|   +--- efs-provisioner-wp.yaml
|   +--- kubernetes_setup.tcl
+--- etcd
|   +--- etcd-cluster.yaml
|   +--- etcd-operator-deployment.yml
+--- ingress_loadbalancer
|   +--- ingress-controller-service.yml
|   +--- ingress-controller.yml
|   +--- ingress.yml
+--- mysql-efs.yaml
+--- wordpress-efs.yaml
```



A) Infrastructure (point 1+2 above)
   - Setup kubernetes cluster on AWS
B) Etcd Deployment (point 3)
   - Deploy etcd in kubebernets cluster.
   - Deploy etcd-operator first, then etcd-cluster
C) Storage Management (Point 4)
   EFS ⇔ PersistentVolumeClaim (**accessModes: ReadWriteMany**) ⇔ mysql application (deployment)

   EFS ⇔ PersistentVolumeClaim (**accessModes: ReadWriteMany**) ⇔ wordpress application (deployment)

D) Network Management – Load Balancer (Point 5)
   <u>Ingress-controller</u>
   - Use nginx-ingress-controller image as deployment
   - With "**default-backend-service**" args → "wordpress"
   - Setup servicesaccount, ClusterRole, nginx-ingress-role & RoleBinding, ClusterRoleBinding in case cluster is using RBAC.

- Ingress-controller-service.yaml will expose ingress controller with ExternalIP used from ELB (**type: LoadBalancer**), expose port 80/443 to the outside world.

Ingress-resource:

- Use “Kind: Ingress”
- Use annotation: nginx.ingress.kubernetes.io/rewrite-target to redirect “cluster.kubes.com/careers” to default page

---

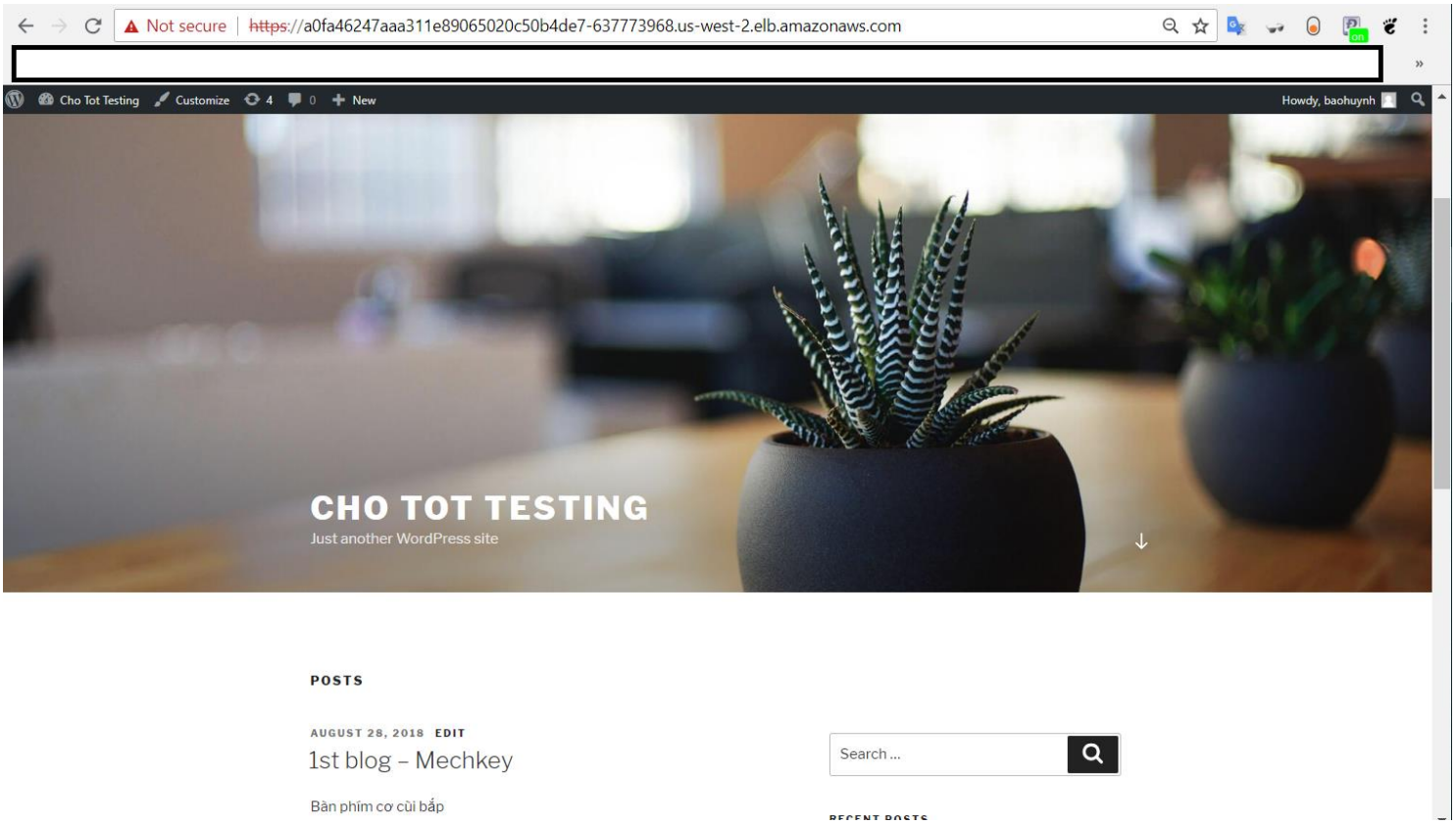
TRAFFICE FLOW:

1) When there’s an request from client, package request will reach <PUBLIC_IP_ALB>:80/443 of ALB AWS first
   ALB check health of nodes where ingress controller belongs to make sure the traffic will be route to alive pods.
2) Package is forwarded to ***Nginx ingress controller***.
3) Ingress controller read “rules” defined in “*ingress resource*” to make routing decision for packet, then route corresponding ***Services*** *(WordPress in this case).*
4) Package is then routed to a node in the service (with correct port-mapping).
5) Package in sent to container inside a pod (with correct port-mapping).

ISSUE DURING IMPLEMENTATION

No	Problems	Investigate & Solution
1	Prior to EFS, I intend to use Portworx for shared volume. But somehow, its app failed to deploy on kube cluster with port refused connection (?). @ip-172-20-47-239 px-runc[24656]: bash: connect: Connection refused @ip-172-20-47-239 px-runc[24656]: bash: /dev/tcp/localhost/9009: Connection refused time="2018-08-25T20:07:55Z" level=warning msg="Could not retrieve PX node status" error="Get http://127.0.0.1:9001/v1/cluster/nodehealth: dial tcp 127.0.0.1:9001: getsockopt: connection refused"	Debug: Check logs from master node: - /var/log/kube-controller-manager.log - kubectl logs <portworx_pod> ---- Portworx requires about 10 ports for their application. Suspect that others app is occupying those ports so I change to port range higher [10000-10010] but nothing changes → EFS seems to be simpler :D
2	- Intend to use DaemonSet to distribute copied of a pod among nodes in cluster as it requires only “Kind:DaemonSet”. - But using DaemonSet make it impossible to expose like “deployment” and use service to create ELB.	Use “Kind: Deployment” with 'antiAffinity' rules to ensure distributed pods among cluster.
3	Implementation of ingress loadbancer costs me a lot of times & efforts. Plenty of examples out there are outdated or malfunctioned with latest update.	- Try & Try & Try & Try example codes. - Read documents about suspected points. - Goolgle is my best-friend.
		Before assignment, my debug skill on Kubernetes is nearly 0. Now, it’s improved a lot ☺

DEMO ☺




```
root@ip-172-20-56-75:~/trial_share_fs/minhho_ex/etcd# kb get ingress,nodes,pods,services,deployment
NAME                                        HOSTS     ADDRESS            PORTS     AGE
ingress.extensions/chotot-redirect-careers  *         a0fa46247aaa3...   80        5h

NAME                                          STATUS    ROLES     AGE       VERSION
node/ip-172-20-45-71.us-west-2.compute.internal    Ready     node      1d        v1.10.3
node/ip-172-20-54-162.us-west-2.compute.internal   Ready     master    1d        v1.10.3
node/ip-172-20-57-226.us-west-2.compute.internal   Ready     node      1d        v1.10.3

NAME                                        READY     STATUS    RESTARTS   AGE
pod/efs-provisioner-6945c9fc7c-6gsrv        1/1       Running   0          1d
pod/efs-provisioner-mysql-5859b4bdd8-vm2qs  1/1       Running   0          1d
pod/etcd-cluster-g7ps7lq4qq                 1/1       Running   0          11m
pod/etcd-cluster-h5pl8hzkf2                 1/1       Running   0          10m
pod/etcd-cluster-ll7jns9mh6                 1/1       Running   0          11m
pod/etcd-operator-69b559656f-rxppk          1/1       Running   0          12m
pod/nginx-ingress-controller-986c64bb8-zds5x 1/1       Running   0          5h
pod/wordpress-d5dd7578-h5p89                1/1       Running   0          5h
pod/wordpress-d5dd7578-r684x                1/1       Running   0          5h
pod/wordpress-mysql-78df659f64-jwbnp        1/1       Running   0          5h

NAME                        TYPE           CLUSTER-IP       EXTERNAL-IP        PORT(S)                    AGE
service/etcd-cluster        ClusterIP      None             <none>             2379/TCP,2380/TCP          11m
service/etcd-cluster-client ClusterIP      100.71.91.32     <none>             2379/TCP                   11m
service/ingress-nginx       LoadBalancer   100.71.154.116   a0fa46247aaa3...   80:30618/TCP,443:32105/TCP 5h
service/kubernetes          ClusterIP      100.64.0.1       <none>             443/TCP                    1d
service/wordpress           ClusterIP      100.71.150.38    <none>             80/TCP                     5h
service/wordpress-mysql     ClusterIP      None             <none>             3306/TCP                   5h

NAME                                        DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deployment.extensions/efs-provisioner       1         1         1            1           1d
deployment.extensions/efs-provisioner-mysql 1         1         1            1           1d
deployment.extensions/etcd-operator         1         1         1            1           12m
deployment.extensions/nginx-ingress-controller 1      1         1            1           5h
deployment.extensions/wordpress             2         2         2            2           5h
deployment.extensions/wordpress-mysql       1         1         1            1           5h
root@ip-172-20-56-75:~/trial_share_fs/minhho_ex/etcd#
```

Create Load Balancer | Actions

Filter by tags and attributes or search by keyword

1 to 2 of 2

Name	DNS name	State
a0fa46247aaa311e8906502…	a0fa46247aaa311e89065020c50b4de7-637773968.us-west-2.elb.amazonaws.com	
webserver-example-qa	webserver-example-qa-1005511570.us-west-2.elb.amazonaws.com	

Connection Draining: Disabled (Edit)

Edit Instances

Instance ID	Name	Availability Zone	Status	Actions
i-06c42f689f5b47737	nodes.cluster.chotot.vpc	us-west-2a	InService	Remove from Load Balancer
i-04f10d2e7bf032b49	nodes.cluster.chotot.vpc	us-west-2a	InService	Remove from Load Balancer

Edit Availability Zones

Availability Zone	Subnet ID	Subnet CIDR	Instance Count	Healthy?	Actions
us-west-2a	subnet-019427bc1d90f1091	172.20.32.0/19	2	Yes	-