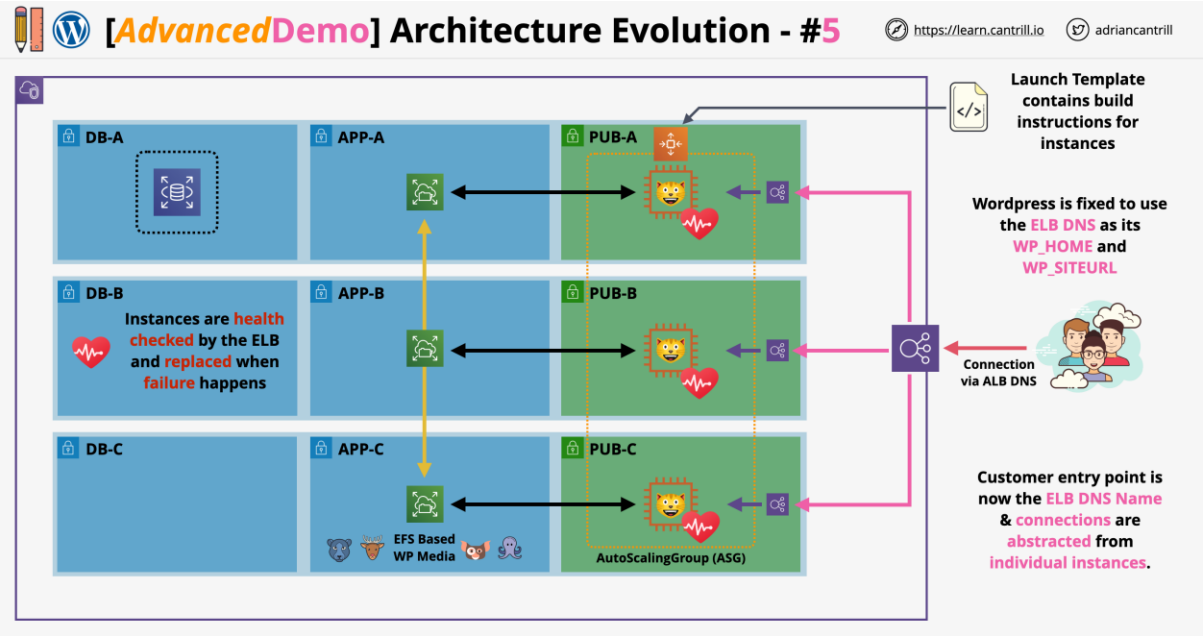
Advanced Demo - Web App - Single Server to Elastic Evolution



In stage 5 of this advanced demo lesson, you will be adding an auto scaling group to provision and terminate instances automatically based on load on the system.

You have already performed all of the preparation steps required, by moving data storage onto RDS, media storage onto EFS and creating a launch template to automatically build the wordpress application servers.

STAGE 5A - Create the load balancer

Move to the EC2 console https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#Home:
Click Load Balancers under Load Balancing
Click Create Load Balancer
Click Create under Application Load Balancer
Under name enter A4LWORDPRESSALB
Ensure internet-facing is selected
ensure ipv4 selected for IP Address type

Under Network Mapping select A4LVPC in the VPC Dropdown
Check the boxes next to us-east-1a us-east-1b and us-east-1c
Select sn-pub-A, sn-pub-B and sn-pub-C for each.

Scroll down and under Security Groups remove default and select the A4LVPC-SGLoadBalancer from the dropdown.

Under Listener and Routing Ensure Protocol is set to HTTP and Port is set to 80.
Click Create target group which will open a new tab
for Target Type choose Instances for Target group name choose A4LWORDPRESSALBTG
For Protocol choose HTTP
For Port choose 80
Make sure the VPC is set to A4LVPC
Check that Protocol Version is set to HTTP1

Under Health checks for Protocol choose HTTP and for Path choose /
Click Next
We wont register any right now so click Create target Group
Go back to the previous tab where you are creating the Load Balancer Click the Refresh Icon and select the A4LWORDPRESSALBTG item in the dropdown.
Scroll down to the bottom and click Create load balancer
Click View Load Balancer and select the load balancer you are creating.
Scroll down and copy the DNS Name into your clipboard

STAGE 5B - Create a new Parameter store value with the ELB DNS name

Move to the systems manager console https://console.aws.amazon.com/systems-manager/home?region=us-east-1#
Click Parameter Store
Click Create Parameter
Under Name enter /A4L/Wordpress/ALBDNSNAME Under Description enter DNS Name of the Application Load Balancer for wordpress
for Tier set Standard
For Type set String
for Data Type set text
for Value set the DNS name of the load balancer you copied into your clipboard Click Create Parameter

STAGE 5C - Update the Launch template to wordpress is updated with the ELB DNS as its home

Go to the EC2 console https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#Home:
CLick Launch Templates
Check the box next to the Wordpress launch template, click Actions and click Modify Template (Create New Version)
for Template version description enter App only, uses EFS filesystem defined in /A4L/Wordpress/EFSFSID, ALB home added to WP Database
Scroll to the bottom and expand Advanced Details
Scroll to the bottom and find User Data expand the entry box as much as possible.

After #!/bin/bash -xe position cursor at the end & press enter twice to add new lines paste in this

```
ALBDNSNAME=$(aws ssm get-parameters --region us-east-1 --names /A4L/Wordpress/ALBDNSNAME --query Parameters[0].Value)

ALBDNSNAME=`echo $ALBDNSNAME | sed -e 's/^"//' -e 's/"$//'`
```

Move all the way to the bottom of the User Data and paste in this block

```
cat >> /home/ec2-user/update_wp_ip.sh<< 'EOF'

#!/bin/bash

source <(php -r 'require("/var/www/html/wp-config.php"); echo("DB_NAME=".DB_NAME."; DB_USER=".DB_USER."; DB_PASSWORD=".DB_PASSWORD."; DB_HOST=".DB_HOST); ')

SQL_COMMAND="mysql -u $DB_USER -h $DB_HOST -p$DB_PASSWORD $DB_NAME -e"
```

OLD_URL=$(mysql -u $DB_USER -h $DB_HOST -p$DB_PASSWORD $DB_NAME -e 'select option_value from wp_options where option_id = 1;' | grep http)

ALBDNSNAME=$(aws ssm get-parameters --region us-east-1 --names /A4L/Wordpress/ALBDNSNAME --query Parameters[0].Value)

ALBDNSNAME=`echo $ALBDNSNAME | sed -e 's/^"//' -e 's/"$//'`

$SQL_COMMAND "UPDATE wp_options SET option_value = replace(option_value, '$OLD_URL', 'http://$ALBDNSNAME') WHERE option_name = 'home' OR option_name = 'siteurl';"

$SQL_COMMAND "UPDATE wp_posts SET guid = replace(guid, '$OLD_URL','http://$ALBDNSNAME');"

$SQL_COMMAND "UPDATE wp_posts SET post_content = replace(post_content, '$OLD_URL', 'http://$ALBDNSNAME');"

$SQL_COMMAND "UPDATE wp_postmeta SET meta_value = replace(meta_value,'$OLD_URL','http://$ALBDNSNAME');"

EOF

chmod 755 /home/ec2-user/update_wp_ip.sh

echo "/home/ec2-user/update_wp_ip.sh" >> /etc/rc.local

/home/ec2-user/update_wp_ip.sh

Scroll down and click Create template version
Click View Launch Template
Select the template again (dont click) Click Actions and select Set Default Version
Under Template version select 4
Click Set as default version

STAGE 5D - Create an auto scaling group (no scaling yet)

Move to the EC2 console
under Auto Scaling
click Auto Scaling Groups
Click Create an Auto Scaling Group
For Auto Scaling group name enter A4LWORDPRESSASG
Under Launch Template select Wordpress
Under Version select Latest
Scroll down and click Next
For Network VPC select A4LVPC
For Subnets select sn-Pub-A, sn-pub-B and sn-pub-C
Click next

STAGE 5E - Integrate ASG and ALB

Its here where we integrate the ASG with the Load Balancer. Load balancers actually work (for EC2) with static instance registrations. What ASG does, it link with a target group, any instances provisioned by the ASG are added to the target group, anything terminated is removed.

Check the Attach to an existing Load balancer box
Ensure Choose from your load balancer target groups is selected.
for existing load balancer targer groups select A4LWORDPRESSALBTG
don't make any changes to VPC Lattice integration options

Under health Checks check Turn on Elastic Load Balancing health checks
Under Additional Settings check Enable group metrics collection within CloudWatch

Scroll down and click Next

For now leave Desired Mininum and Maximum at 1
For Scaling policies - optional leave it on None
Make sure Enable instance scale-in protection is **NOT** checked
Click Next
We wont be adding notifications so click Next Again
Click Add Tag
for Key enter Name and for Value enter Wordpress-ASG make sure Tag New instances is checked
Click Next Click Create Auto Scaling Group

Right click on instances and open in a new tab
Right click Wordpress-LT, Terminate Instance and confirm.
This removes the old manually created wordpress instance Click Refresh and you should see a new instance being created... Wordpress-ASG this is the one created automatically by the ASG using the launch template - this is because the desired capacity is set to 1 and we currently have 0

STAGE 5F - Add scaling

Move to the AWS Console https://console.aws.amazon.com/ec2/autoscaling/home?region=us-east-1#AutoScalingGroups:
Click Auto Scaling Groups
Click the A4LWORDPRESSASG ASG
Click the Automatic SCaling Tab

We're going to add two policies, scale in and scale out.

SCALEOUT when CPU usage on average is above 40%

Click Create dynamic Scaling Policy
For policy type select Simple scaling
for Scaling Policy name enter HIGHCPU
Click Create a CloudWatch Alarm
Click Select Metric
Click EC2
Click By Auto Scaling Group (if this doesn't show in your list, wait a while and refresh)
Check A4LWORDPRESSASG CPU Utilization (if this doesn't show in your list, wait a while and refresh)
Click Select Metric
Scroll Down... select Threashhold style static, select Greater and enter 40 in the than box and click Next Click Remove next to notification if you see anything listed here

Click Next Enter WordpressHIGHCPU in Alarm Name
Click Next
Click Create Alarm
Go back to the AutoScalingGroup tab and click the Refresh SYmbol next to Cloudwatch Alarm
Click the dropdown and select WordpressHIGHCPU
For Take the action choose Add 1 Capacity units
Click Create

SCALEIN when CPU usage on average ie below 40%

Click Create Dynamic Scaling Policy
For policy type select Simple scaling
for Scaling Policy name enter LOWCPU
Click Create a CloudWatch Alarm
Click Select Metric
Click EC2
Click By Auto Scaling Group Check A4LWORDPRESSASG CPU Utilization
Click Select Metric
Scroll Down... select Static, Lower and enter 40 in the than box and click next Click Remove next to notification Click Next Enter WordpressLOWCPU in Alarm Name
Click Next
Click Create Alarm
Go back to the AutoScalingGroup tab and click the Refresh Symbol next to Cloudwatch Alarm
Click the dropdown and select WordpressLOWCPU
For Take the action choose Remove 1 Capacity units
Click Create

ADJUST ASG Values

Click Details Tab
Under Group Details click Edit
Set Desired 1, Minimum 1 and Maximum 3
Click Update

STAGE 5G - Test Scaling & Self Healing

Open Auto Scaling Groups in a new tab https://console.aws.amazon.com/ec2autoscaling/home?region=us-east-1#/details
Open that Auto scaling group in that tab and click on Activity tab Go to running instances in the EC2 Console https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#Instances:sort=desc:tag:Name in a new tab

Simulate some load on the wordpress instance

Select the/one running Wordpress-ASG instance, right click, Connect, Select Session Manager and click Connect
type sudo bash and press enter
type cd and press enter
type clear and press enter

run stress -c 2 -v -t 3000

this stresses the CPU on the instance, while running go to the ASG tag, and refresh the activities tab. it might take a few minutes, but the ASG will detect high CPU load and begin provisioning a new EC2 instance. if you want to see the monitoring stats, change to the monitoring tag on the ASG Console Check enable next to Auto Scaling Group Metrics Collection

At some point another instance will be added. This will be auto built based on the launch template, connect to the RDS instance and EFS file system and add another instance of capacity to the platform. Try terminating one of the EC2 instances ... Watch what happens in the activity tab of the auto scaling group console. This is an example of self-healing, a new instance is provisioned to take the old ones place.

STAGE 5 - FINISH

This configuration has several limitations :-

- ~~The application and database are built manually, taking time and not allowing automation~~ FIXED
- ~~^^ it was slow and annoying ... that was the intention.~~ FIXED
- ~~The database and application are on the same instance, neither can scale without the other~~ FIXED
- ~~The database of the application is on an instance, scaling IN/OUT risks this media~~ FIXED
- ~~The application media and UI store is local to an instance, scaling IN/OUT risks this media~~ FIXED
- ~~Customer Connections are to an instance directly ... no health checks/auto healing~~ FIXED
- ~~The IP of the instance is hardcoded into the database ....~~ FIXED

You can now move onto STAGE6 which is the cleanup step.