WordPress Deployment on AWS

# Objectives

To demonstrate how infrastructure provisioning and code deployment can be automated on AWS. Following AWS cloud services are used for this demo:

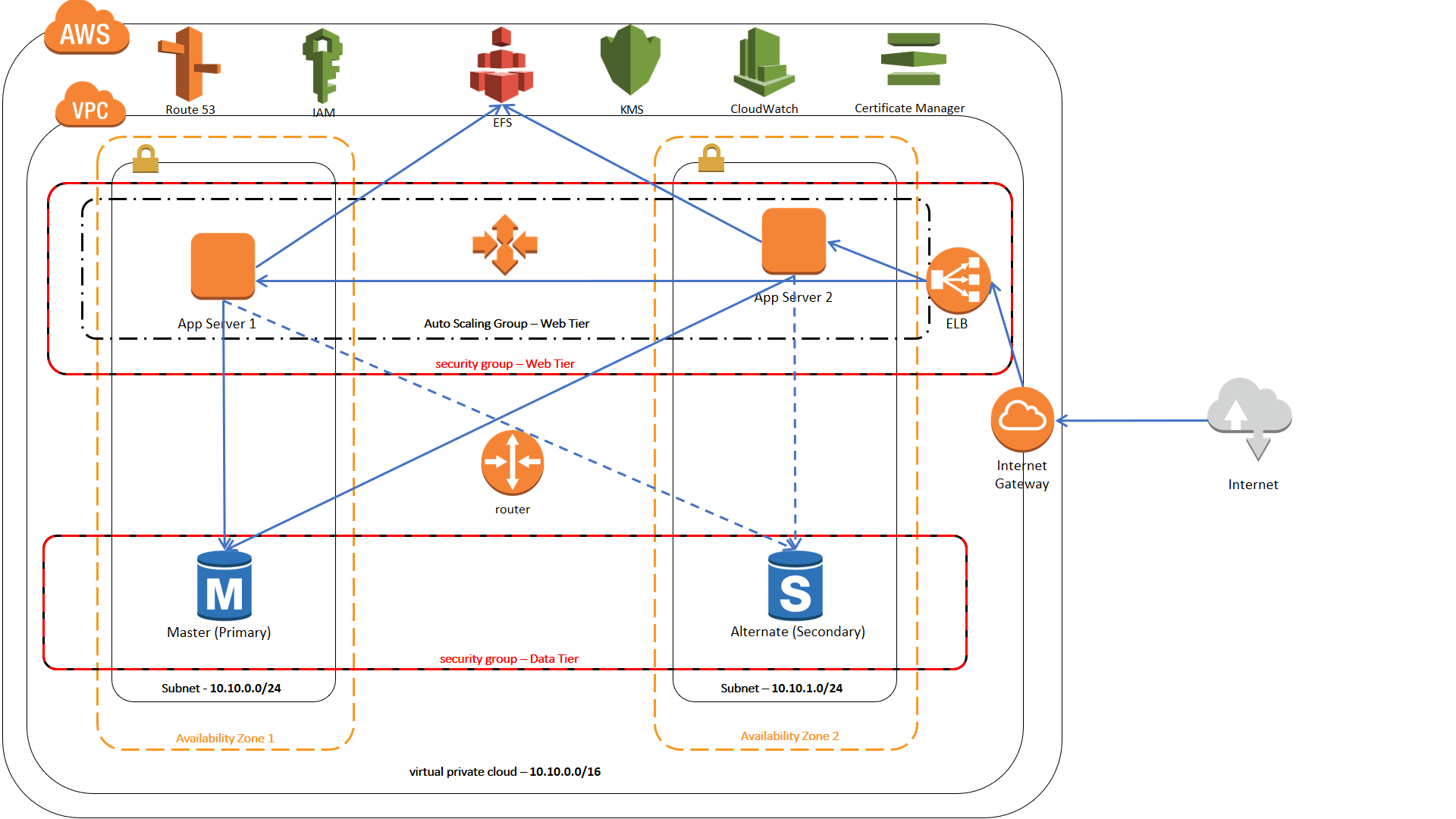
* Networking: VPC, RouteTable, Subnet, SecurityGroup, ACL
* Compute: EC2, Elastic Load Balancer, AutoScaling
* Storage: EBS, EFS
* Database: RDS (MySQL)
* Security: IAM, KMS, ACM
* DevOps: CodeDeploy, CloudFormation
* Monitoring: CloudWatch

# Approach

* WordPress is chosen for this demo
* A copy of WordPress has been downloaded from wordpress.org and is stored on a Github public repository called cloudsmart/mywordpress
* Deployment scripts used by CodeDeploy are stored in a directory called scripts
* Infrastructure provisioning templates used by CloudFormation are stored in a directory called infrastructure

# Architecture Highlights

* Two-tier application comprised of:
  + Web Tier: Includes two EC2 instances spread across two Availability Zones. The EC2 instances are behind a classic load balancer (ELB) and are associated with an Auto Scaling Group to scale them up or down based on their CPU usage that’s being monitored by CloudWatch.
    - Adds more instances (to the maximum of four instances) if CPU usage stays at or above 75% for five consecutive minutes
    - Removes extra instances (keeps minimum two instances) if CPU usage stays at 25% or lower for five consecutive minutes
  + Data Tier: Includes MySQL database in RDS Multi-AZ configuration for high availability and fault tolerance
* EFS is used as a shared file store (NAS) among EC2 instances to hold the media content (image, video, etc.) uploaded by users to WordPress
* There are three Security Groups:
  + Default: Allows all sort of communication between services inside the VPC, e.g. from EC2 instances to EFS
  + Web Tier: Allows inbound traffic on HTTP:80, HTTPS:443, SSH:22 from the Internet.
    - Note: SSH port 22 is open to the Internet intentionally for easier access since this is a demo. In real world scenarios it mush be locked down to specific IP address or IP range
  + Data Tier: Allows inbound traffic on MySQL:3306 from Web Tier security group only

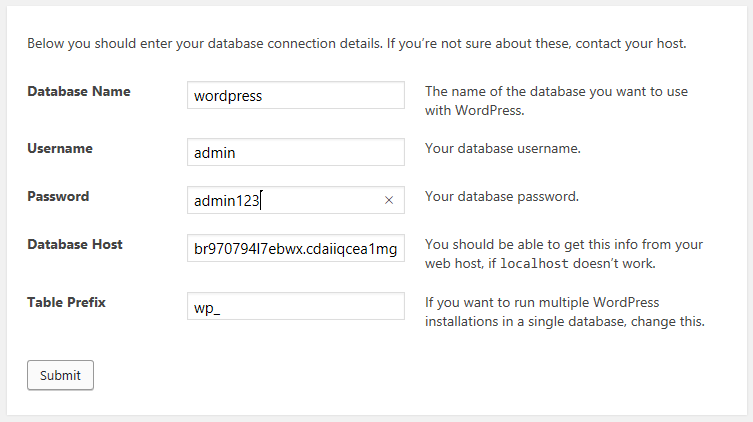


# Deployment Procedure

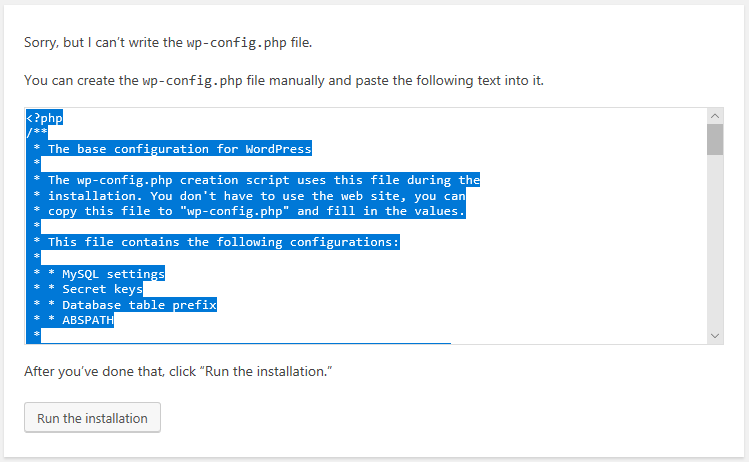
* Infrastructure
  + Make a local copy of the WordPress.template file located under cloudsmart/mywordpress/scripts/cloudformation
  + Create a new stack in CloudFormation and upload the file through “Upload a template to Amazon S3” and hit next
  + Provide a name for the stack (e.g. WordPress-Stack)
  + Provide input to the parameters:
    - AsgMaxSize: Maximum size of Auto Scaling Group, default is 4
    - AsgMinSize: Minimum size and initial desired size of Auto Scaling Group, default is 2
    - AZ1: Availability Zone 1 (prepopulated list based on the current region)
    - AZ2: Availability Zone 2 (prepopulated list based on the current region)
    - DigitalCertificate: Digital Certificate Identifier for SSL. You need to issue a Certificate through ACM prior to this step
    - KeyPair: KeyPair used to SSH to EC2 instances (values get populated based on the current region). You need to create a Key pair prior to this step
    - MySQLMasterUserName: UserName for the MySQL master (power) user
    - MySQLMasterUserPassword: Password for the MySQL master (power) user
  + Run the stack to provision and configure the infrastructure
* Code Deployment
  + Create a new Application under CodeDeploy console and provide the following inputs when requested:
    - Application Name: e.g. WordPress-App
    - Compute Platform: EC2/On-Premises (default)
    - Deployment Group Name: e.g. WordPress-Deployment
    - In-pace deployment: should be selected
    - Environment Configuration: Enter following information under Amazon EC2 Instances to select the proper EC2 instances for deployment:
      * Key: Name
      * Value: BJ-ASG-EC2-WordPress
    - Enable Load Balancer: Check the checkbox, select Classic Lad Balancer and select the item from the dropdown that has a name starting with BJ-CFS-…
    - Deployment Configuration: Select CodeDeployDefault:HalfAtATime
    - Service Role ARN: select the role that is closest to this
      * BJ-CFS-WordPress-…iamRoleCodeDeploy…
  + Select the newly created application and select Deploy New Revision from the Actions menu
  + Provide following information in the Create Deployment page:
    - Repository Type: Select My application is stored in Github
    - Github account: enter the Github account hosting the WordPress repository (e.g. cloudsmart) and hit Connect to Github to connect to the Github account
    - Repository Name: enter the repository name, e.g. cloudsmart/mywordpress
    - Commit ID: enter the Commit ID of the revision you want to deploy (usually the last commit)
    - Content Options: select Overwrite the content
    - Hit deploy
* Remarks:
  + Run one Infrastructure deployment first and then as many Code Deployment as needed
  + Infrastructure deployment creates a brand-new environment from scratch so any previous data in RDS and EFS will be lost

# Launching WordPress

* You can launch WordPress through any Web browser by using the public DNS name of the load balancer created as part of the infrastructure deployment, e.g. http://myloadbalancer.elb.amazonaws.com/
  + Note: Since wordpress stores admin URL’s, https protocol is not working on this site despite being configured on load balancer. To verify, you can use https with the public DNS of the load balancer to access the healthcheck.html page on the server, e.g. https://myloadbalancer.elb.amazonaws.com/healthcheck.html
* First time you launch WordPress it will ask you a few questions to configure its connectivity to the database:
  + Database Name: Go with the default value as wordpress
  + UserName: Enter the database username, same username as provided for infrastructure provisioning in CloudFormation
  + Password: Enter the password, same password as you entered for infrastructure provisioning in CloudFormation
  + Database host: Copy the Database Endpoint from RDS dashboard of the MySQL instance hosting wordpress database and paste it in this box
  + Table Prefix: Go with the default value, wp\_
* After entering all information hit submit to proceed



* Wordpress then provides a script that contains the above information plus additional configuration data which you should copy and paste in a new file called wp-config.php under the /var/www/html folder of both EC2 instances where the rest of wordpress source code resides.
  + Note: Since the Database Host (endpoint) gets changed everytime we build the infrastructure stack, the wp-config.php file is not included in the Github repo and is not deployed through CodeDeploy. For future enhancements, the file can be created through automation, e.g. Lambda function and Custom Resource in CloudFormation, to eliminate manual intervention.
* After the config files are created, hit the Run the Installation button:



* You’d be asked to provide a name for your site as well as your wordpress username and password in the next page. After you provide information and hit install your wordpress application is fully configured and is ready to use
* After this point, all subsequent visits to the website will put you on the landing page of wordpress site where you can view, edit, and share your posts