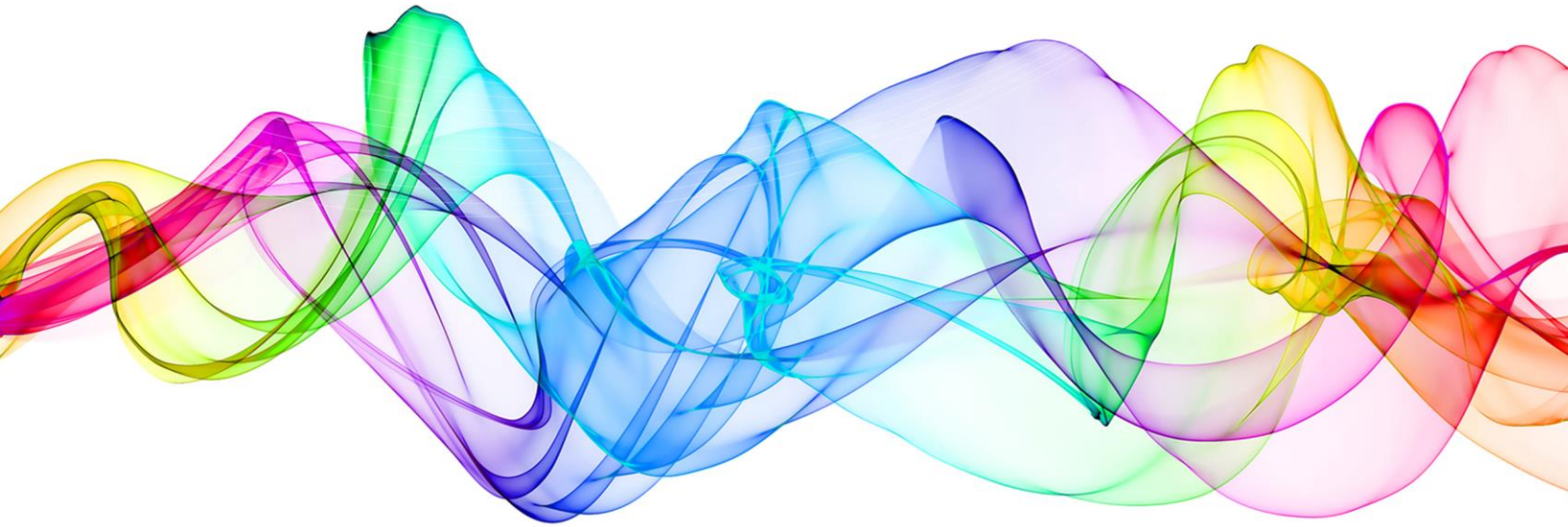




ARCHITECTING YOUR FIRST ENTERPRISE CLASS WEB APPLICATION USING AMAZON WEB SERVICES

A Technical Whitepaper for Cloud Based Development



To Architect and Build an Enterprise Class Application that will be Highly
Available, Extremely Secure, Very Reliable and Incredibly Resilient

Biju Chandrasekharan and Sreejith Nair

Introduction

Stemming from the need to move from CapEx to OpEx, Flexibility and Agility in IT Resource planning and Improve Time to Market, organizations of all sizes are flocking to the Cloud, making it the "new normal" for IT. Gartner has ranked Amazon Web Services as the leader in its latest magic quadrant for Cloud Infrastructure as a service, Worldwide. Among the most popular IaaS providers, AWS has the most functionalities, has the largest customer and partner community and also the longest experience (9 Years). From startups to enterprises to public sectors, AWS has over 1million customers. Security and Compliance used to be the biggest concern for Cloud adoption however, AWS has changed that game. Today, organizations are adopting AWS to become more secure and compliant. Its innovation culture is so strong that in 2015 alone, AWS launched a total of 522 features! Its global infrastructure spans across 5 Continents, 11 Regions, 30 availability zones and 53 edge locations providing 60 enterprise class services offering in categories such as Compute, Storage and Content Delivery, Database, Networking, Developer Tools, Management Tools, Security and Identity, Analytics, Internet of things, Mobile Services, Application Services and Enterprise Applications. With these many options and comprehensive list of services and offerings, it can become very challenging to get started in the right way. Hence, the objective of this paper is to provide you step by step instructions to architect and build an enterprise-class application that will be highly available, extremely secure, very reliable and incredibly resilient. By the time you are finished reading this paper, you will have your own application up and running in the cloud! Sounds exciting? Let's go!

The Problem

The biggest hurdle that developers and architects face during cloud adoption is “How to get started and build something that enterprise-class that is simple to understand and implement yet does not comprise the basics of Enterprise software needs”? There are three key problems that stand out in crossing this hurdle:

- 1) There are several artifacts available today that address important aspects in silos like Security, Infrastructure, Compute, Storage, Network, Database, etc. However, there is a lack of quality artifacts that puts all these pieces together to build a comprehensive solution together, from a technical perspective.
- 2) Most technical papers focus on the “what” part of the solution before the “Why” part of the solution. Without understanding the “Why,” “What” is not important. For example, unless we understand and establish why AWS is important and “why” it is relevant to IT organizations, “What” can be achieved on AWS does not make much sense.

The Solution

Key Design Considerations

AWS is Infrastructure as a Service (IaaS) not a Datacenter

AWS is a very mature Infrastructure as a Service. Just in their core services offerings, AWS offers several options for Compute, Storage, Database, Network and Content Delivery. Hence, it is imperative that a solution built for the cloud understands these core services. Besides the core infrastructure, there are several other services that AWS offers. Under its Management Tools, AWS offers Cloud Watch, which can be combined with one or more of the services such as Simple Notification Services, Load Balancers, Alarms, AMIs, Launch Configuration and Auto Scaling group to provide Automatic Scaling (Scale up/Scale Down) for Application or Services. Amazon's Cloud Trail can be utilized to track user activity and API Usage. AWS has its own Global Content Delivery Network called Cloud front that utilizes one of the 53 Edge Location for Delivering web and RTMP distribution (For streaming media Files). Another popular offering is AWS Route 53, which is a scalable DNS and Domain Name Registration service offering.

Design Principles are different for Data Center and AWS

Since AWS is not a typical Data Center, because what worked well in a Data Center based environment may not function well on the cloud. Here is a quick example of how the selection of virtual servers.

	Data Center	AWS
Compute	Virtualized Servers	Virtualized Servers optimized for General Purposes (M4), Compute Optimized (C4), Memory Optimized (R3), GPU Optimized (G2), Storage Optimized (D2), IO Optimized (I2), Low-Cost Burstable Performance (T2) and Dedicated Instances
Compute – Purchase Options	Allocated for a fixed period, typically in months. Lead time required (6-8 Weeks)	Reserved Instances (1-3 years) On-Demand Instances Spot Instances No Lead time

The table above shows the various options available on AWS that typically is not available in a Data Center due to these differences:

- 1) An Application that needs a lot of graphics should be designed with GPU Optimized (G2) instance type.
- 2) An application that does a lot of batch processing has to be designed differently on AWS to leverage the low cost to leverage the benefits of spot instances

For more on best practices for Architecting, please check out the Best Practices document [here](#).

Try to know what is relevant and important in your context

Why? Well, there are two key reasons. First of all, it is close to impossible to know all the latest and greatest offerings that are coming from AWS. Since its inception in 2006, AWS has been constantly evolving and innovating at a rapid pace to support virtually any kind of workloads. Today, they have over 60 services with 1696 features, out of which 522 new features were just added in 2015.

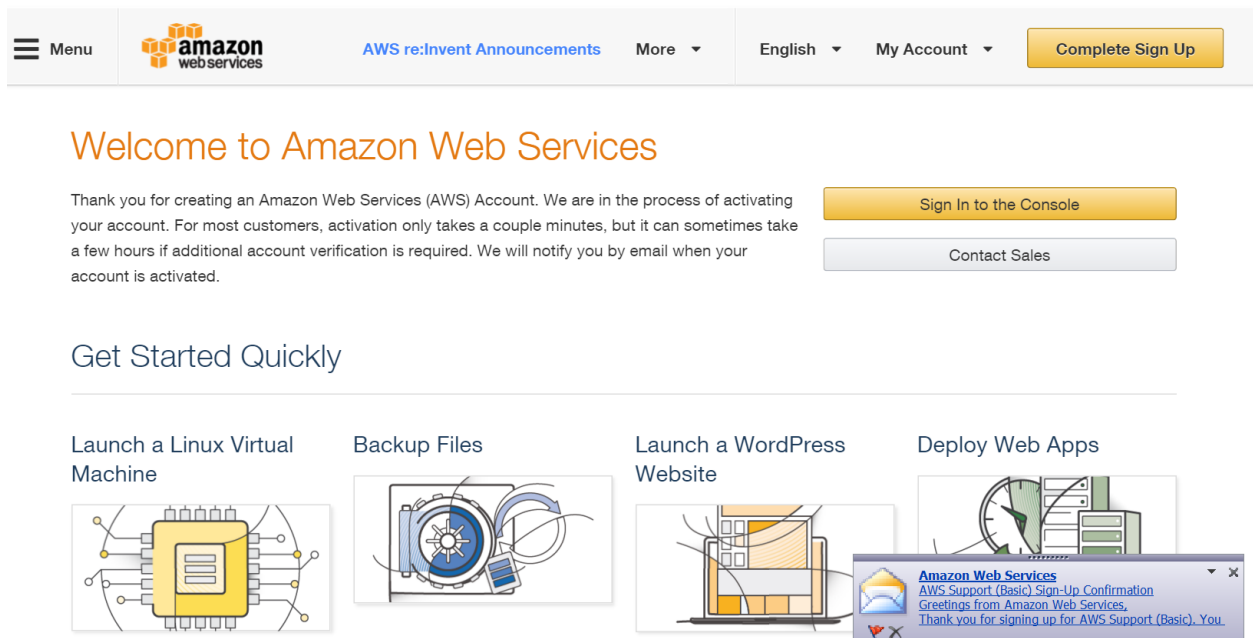
Secondly, these services range from compute, storage, networking, database, analytics, application services, deployment, management, and mobile. An architect responsible for designing and implementing an e-commerce web-based applications **may not** need to know what the new offerings are in analytics and mobility. Having said that, keeping a high-level knowledge may certainly help. Again, this consideration is only for someone trying to get started.

Setting up the AWS Account

Go to <http://aws.amazon.com/> and start the sign-up process. All you need is an email address, a credit card, and a touch phone to verify your account. The sign-up is very simple takes 5 steps which take less than 5minutes:


- 1) Contact information
- 2) Payment Information
- 3) Identity Verification
- 4) Support Plan – This is where you select the support plan. Pick free tier.
- 5) Conformation

You can get this done in less than 5minutes. Once you complete the sign-up, you will see the screen below. You will also get email notifications highlighting the details of the account.



Clicking on the sign in to the console button will take you to the AWS Console.





The AWS Dashboard

 **AWS** ▾ **Services** ▾ **Edit** ▾







Biju Chandrasekharan ▾ Oregon ▾ Support ▾

Amazon Web Services




Compute

-  **EC2**
Virtual Servers in the Cloud
-  **EC2 Container Service**
Run and Manage Docker Containers
-  **Elastic Beanstalk**
Run and Manage Web Apps
-  **Lambda**
Run Code in Response to Events








Storage & Content Delivery

-  **S3**
Scalable Storage in the Cloud
-  **CloudFront**
Global Content Delivery Network
-  **Elastic File System** **PREVIEW**
Fully Managed File System for EC2
-  **Glacier**
Archive Storage in the Cloud
-  **Import/Export Snowball**
Large Scale Data Transport
-  **Storage Gateway**
Integrates On-Premises IT Environments with Cloud Storage


Developer Tools

-  **CodeCommit**
Store Code in Private Git Repositories
-  **CodeDeploy**
Automate Code Deployments
-  **CodePipeline**
Release Software using Continuous Delivery






Management Tools

-  **CloudWatch**
Monitor Resources and Applications
-  **CloudFormation**
Create and Manage Resources with Templates
-  **CloudTrail**
Track User Activity and API Usage
-  **Config**
Track Resource Inventory and Changes
-  **OpsWorks**
Automate Operations with Chef
-  **Service Catalog**
Create and Use Standardized Products
-  **Trusted Advisor**
Optimize Performance and Security




Internet of Things

-  **AWS IoT** **BETA**
Connect Devices to the cloud

Mobile Services

-  **Mobile Hub** **BETA**
Build, Test, and Monitor Mobile apps
-  **Cognito**
User Identity and App Data Synchronization
-  **Device Farm**
Test Android, Fire OS, and iOS apps on real devices in the Cloud
-  **Mobile Analytics**
Collect, View and Export App Analytics
-  **SNS**
Push Notification Service

Application Services

-  **API Gateway**
Build, Deploy and Manage APIs
-  **AppStream**
Low Latency Application Streaming
-  **CloudSearch**
Managed Search Service

Resource Groups

A resource group is a collection of resources that share one or more tags. Create a group for each project, application, or environment in your account.

[Create a Group](#) [Tag Editor](#)

Additional Resources

[Getting Started](#) [Read our documentation](#) or view our [training](#) to learn more about AWS.

[AWS Console Mobile App](#) [View your resources on the go with our AWS Console mobile app, available from Amazon Appstore, Google Play, or iTunes.](#)

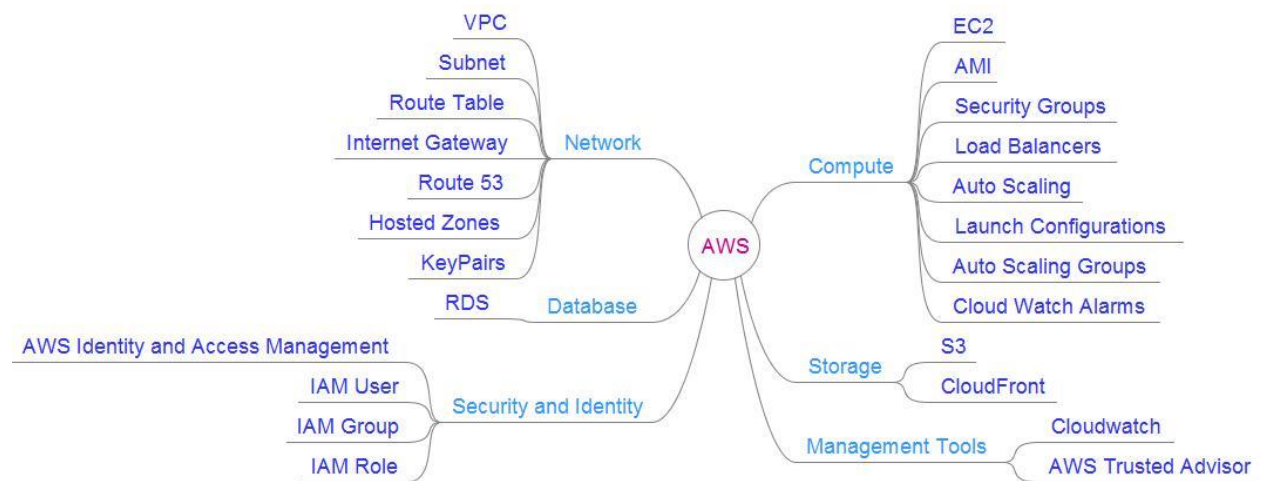
[AWS Marketplace](#) [Find and buy software, launch with 1-Click and pay by the hour.](#)

Yay! Welcome to the world of Cloud Computing. You are all set to explore Amazon Web Service!

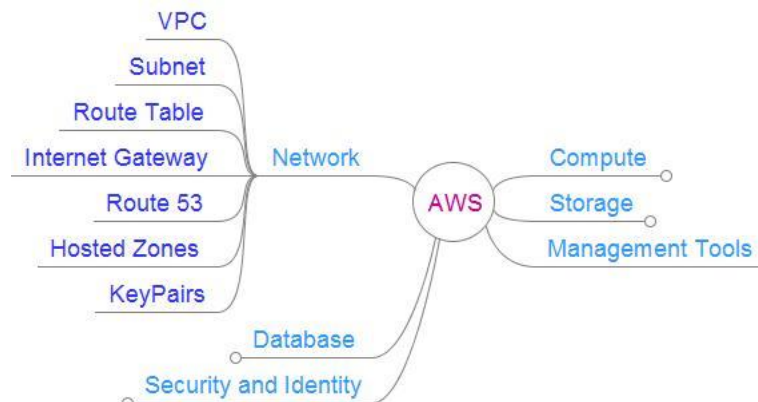
Congratulations on getting to this point!

Understanding the Key Services

Before we go on to building the application, it is important to get familiarized with the all the services that we will use. If you are not new to AWS, you can skip this section or skim through it. It will come in handy as we assemble the solution using these building blocks. We have selected only the important things that you need to know. My no means this is a comprehensive definition of these services.



Network



VPC: Virtual Private Cloud (Amazon VPC) is a logically isolated section of the Amazon Web Services (AWS) that you can define within which you can launch your AWS resources. Within this virtual networking environment, you have complete control to select your own IP address range, subnet creation, configuring route tables and internet gateways. VPC allows you to create a public-facing subnet for your web servers that has access to the Internet and place your databases or application servers in a private-facing subnet with no Internet access. You can leverage multiple layers of security like security groups and network access control lists to help control access to Amazon EC2 instances in each subnet. Additionally, you can create a Hardware Virtual Private Network (VPN) connection between your corporate datacenter and your VPC and leverage the AWS cloud as an extension of your corporate datacenter.

Subnet: Subnet is a segment of a VPC's IP address range where you can place groups of isolated resources.

Route Tables: A route table contains a set of rules, called routes that are used to determine where network traffic is directed.

Internet Gateway: Internet Gateway is the Amazon VPC side of a connection to the public internet.

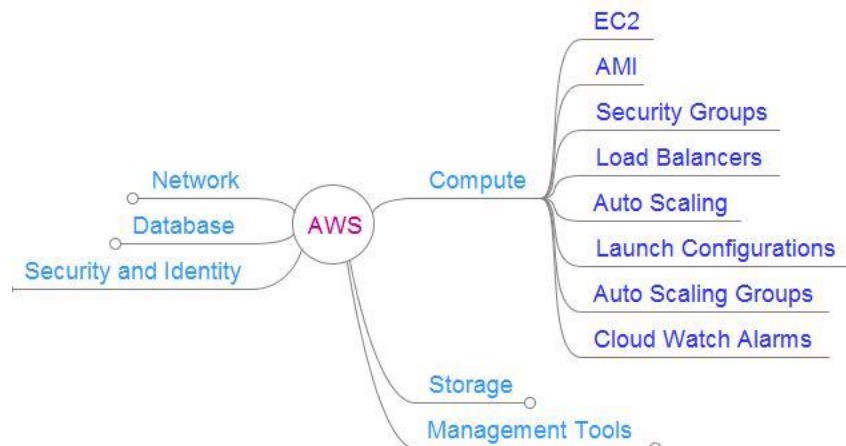
Route 53: Amazon Route 53 lets you register domain names such as example.com. Also, Amazon Route 53 resolved domains names like www.example.com into IP addresses like 192.0.2.1. Amazon

Route 53 responds to DNS queries using a global network of authoritative DNS servers, which reduces latency.

Hosted Zones: A Hosted Zone is a collection of resource record sets hosted by Amazon Route 53. Like a traditional DNS zone file, a hosted zone represents a collection of resource record sets that are managed together under a single domain name.

KeyPairs: Amazon EC2 uses public–key cryptography to encrypt and decrypt login information. Public–key cryptography uses a public key to encrypt a piece of data, such as a password, and then the recipient uses the private key to decrypt the data. The public and private keys are known as a key pair.

Compute



EC2: Amazon EC2 provides scalable, on-demand and OpEx based computing capacity. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage.

AMI: An Amazon Machine Image (AMI) provides the information required to launch an EC2 instance. You specify an AMI when you launch an instance, and you can launch as many instances from the AMI as you need. You can also launch instances from as many different AMIs as you need.

Security Groups: A security group acts as a virtual firewall that controls the traffic for one or more instances. When you launch an instance, you associate one or more security groups with the instance. You add rules to each security group that allows traffic to or from its associated instances. You can modify the rules for a security group at any time; the new rules are automatically applied to all instances that are associated with the security group.

Load Balancers: Elastic Load Balancing automatically distributes incoming application traffic across multiple Amazon EC2 instances in the cloud. It enables you to achieve greater levels of fault tolerance in your applications, seamlessly providing the required amount of load balancing capacity needed to distribute application traffic.

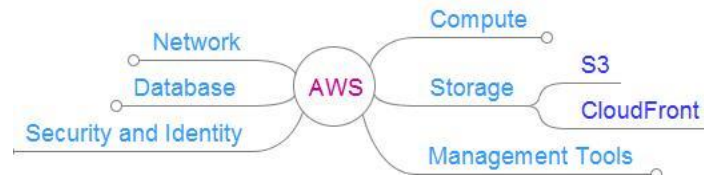
Auto Scaling: Auto Scaling helps you maintain application availability and allows you to scale your Amazon EC2 capacity up or down automatically according to conditions you define. You can use Auto Scaling to help ensure that you are running your desired number of Amazon EC2 instances.

Launch Configurations: A launch configuration is a template that an Auto Scaling group uses to launch EC2 instances. When you create a launch configuration, you specify information for the instances such as the ID of the Amazon Machine Image (AMI), the instance type, a key pair, one or more security groups, and a block device mapping.

Auto-scaling groups: An Auto Scaling group contains a collection of EC2 instances that share similar characteristics and are treated as a logical grouping, for instance scaling and management. For example, if a single application operates across multiple instances, you might want to increase the number of instances in that group to improve the performance of the application, or decrease the number of instances to reduce costs when demand is low. You can use the Auto Scaling group to scale the number of instances automatically based on criteria that you specify, or maintain a fixed number of instances even if an instance becomes unhealthy. This automatic scaling and maintaining the number of instances in an Auto Scaling group is the core value of the Auto Scaling service.

CloudWatch Alarm: A CloudWatch alarm watches a single metric (For Example CPU or IOPS etc.) over a period you specify, and performs one or more actions based on the value of the metric relative to a given threshold over a number of time periods. The action is a notification sent to an Amazon Simple Notification Service topic or Auto Scaling policy. Alarms invoke actions for sustained state changes only.

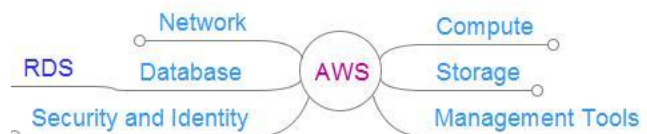
Storage



Amazon Simple Storage Service (Amazon S3): S3 provides developers and IT teams with secure, durable, highly scalable object storage. Amazon S3 offers a range of storage classes designed for different use cases including Amazon S3 Standard for general-purpose storage of frequently accessed data, Amazon S3 Standard - Infrequent Access (Standard - IA) for long-lived, but less frequently accessed data, and Amazon Glacier for long-term archive. Amazon S3 also offers configurable lifecycle policies for managing your data throughout its lifecycle.

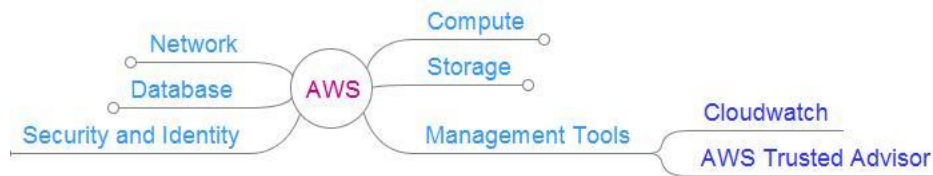
CloudFront: Amazon CloudFront is a content delivery web service. It integrates with other Amazon Web Services products to give an easy way to distribute content to end users with low latency, high data transfer speeds, and no minimum usage commitments.

Database



Amazon Relational Database Service (Amazon RDS): RDS makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while managing all administrative tasks. Amazon RDS provides you six familiar database engines to choose from, including Amazon Aurora, Oracle, Microsoft SQL Server, PostgreSQL, MySQL, and MariaDB.

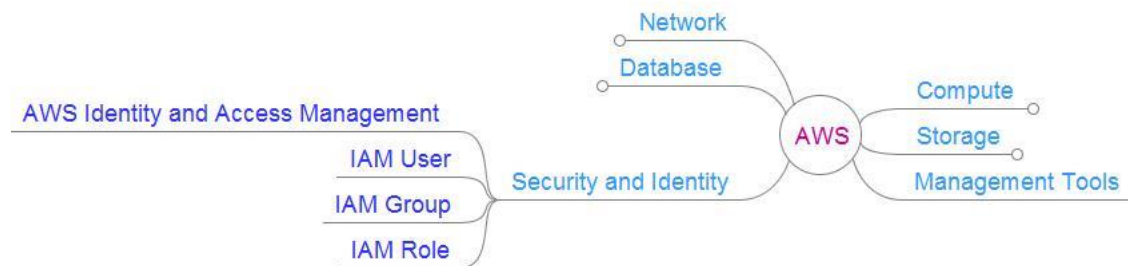
Management Tools



Amazon CloudWatch: CloudWatch is a monitoring service for AWS cloud resources and the applications you run on AWS. You can use Amazon CloudWatch to collect and track metrics, collect and monitor log files, and set alarms. Amazon CloudWatch can monitor AWS resources such as Amazon EC2 instances, Amazon DynamoDB tables, and Amazon RDS DB instances, as well as custom metrics generated by your applications and services, and any log files your applications generate. You can use Amazon. CloudWatch to gain system-wide visibility into resource utilization, application performance, and operational health.

AWS Trusted Advisor: Trusted Advisor inspects your AWS environment and finds opportunities to save money, improve system performance and reliability, or help close security gaps. Since 2013, customers have viewed over 2.6 million best-practice recommendations and realized over \$350 million in estimated cost reductions.

Security and Identity



AWS Identity and Access Management (IAM): IAM enables you to control access to AWS services and resources for your users. Using IAM, you can create and manage AWS users and groups, and use permissions to allow and deny their access to AWS resources.

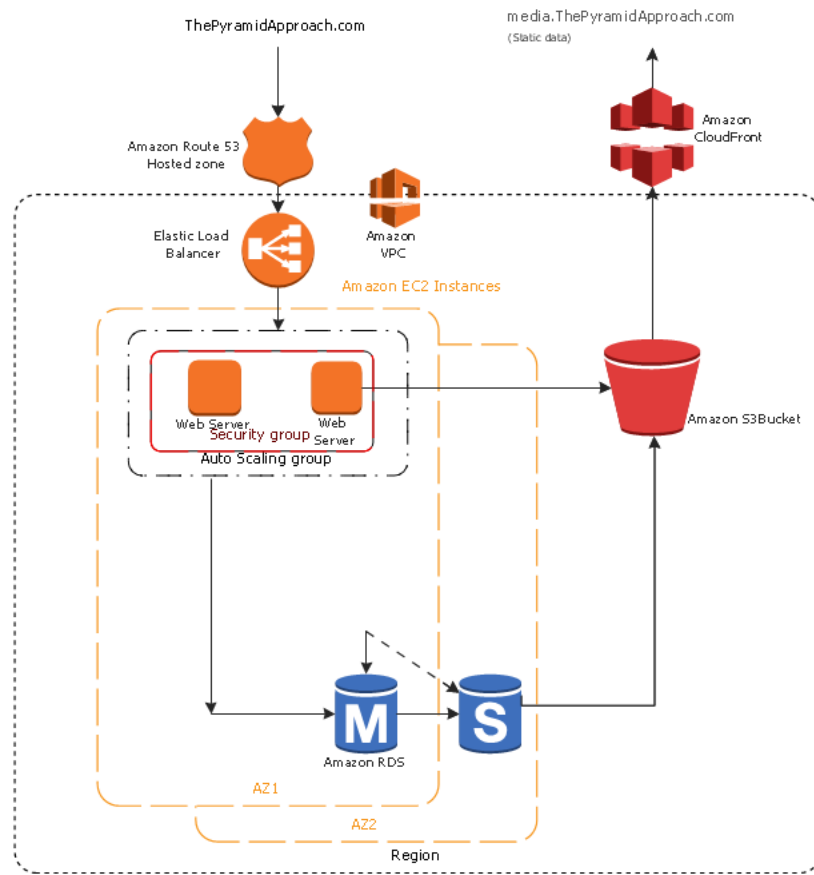
IAM Users: An IAM user is an entity that you create in AWS to represent the person or service using it when interacting with AWS. A primary use for IAM users is to give people you work with the ability to sign in to the AWS Management Console for interactive tasks and to make programmatic requests to AWS services using the API or CLI. A user in AWS consists of a name, a password to sign into the AWS Management Console, and up to two access keys that can be used with the API or CLI.

IAM Groups: An IAM group is a collection of IAM users. You can use groups to specify permissions for a collection of users, which can make those permissions easier to manage for those users. For example, you could have a group called admins and give that group the types of permissions that administrators typically need.

IAM Roles: An IAM role is very similar to a user, in that it is an identity with permission policies that determine what the identity can and cannot do in AWS. However, a role does not have any credentials (password or access keys) associated with it. Instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. An IAM user can assume a role to temporarily take on different permissions for a specific task.

Integrating the Services into a Solution

Let's begin with an end in mind. When we are done, the architecture of the application will look like this.



Step 1: Create a role

- 1) Provide a role name: `rl_S3Access`
- 2) Select AWS Service Role and “Select” Amazon EC2 (Allows EC2 instances to call AWS services on your behalf.)
- 3) Filter By “Policy Type” and select “AmazonS3FullAccess.”
- 4) Finish the process!!

Step 2: Create VPC

- 1) From the Dashboard, Under Networking, Select VPC. DO NOT click on the “Start VPC Wizard”. The reason is that you want to learn to build it from scratch and not use any wizards. CLICK On “Your VPCs” on the left-hand navigation pane and then click on “Create VPC.”
 - a. Name Tag – Provide a name tag like “vpcThePyramid.”
 - b. Please note that you cannot create a VPC larger than /16
 - i. E.g. : 10.0.0.0/16
 - c. Tenancy: Please select “Default”. Setting the tenancy of a VPC to “dedicate” when the VPC is created will ensure that all instances launched in the VPC will run on single-tenant hardware. The tenancy of a VPC cannot be changed after it has been created.
- 2) Route Table: Please click on the “Route Table” on the navigation panel and notice that a new Route Table has been created automatically for the new VPC.

Step 3: Create Subnets

As explained earlier, Subnet is a segment of a VPC's IP address range where you can place groups of isolated resources. We will define 4 subnets here. 2 for the Web Servers and 2 for the DB server.

The key thing to remember here is that a subnet is specific to an Availability Zone (AZ). Hence, it is very important to define at least 2 AZ's for each tier to improve reliability. Again, creating a Subnet is rather simple.

- 1) Provide a name 'Tag: Provide something like snDBThePyramid-10.0.40.0-us-west-2c where "sn" is the prefix, "DB" says that you will use it for your DB Servers, "ThePyramid" is the name of the Application, "10.0.40.0" tells you the IP range and "Us-west-2c" tells you the AZ. You can name it based on your best practice.
- 2) VPC – Select the VPC you just created
- 3) Availability Zone: Pick the Availability Zone you want the subnet to be for. Please note that you cannot have 1 subnet across multiple AZ's
- 4) CIDR Block: Define the CIDR block for the Subnet based on the VPC. For example, it could be 10.0.40.0/24.

For the Database Subnets - ONLY

- 1) Go back to the main dashboard and Click on RDS. On the navigation pane, select subnet groups. Click "DB Subnet Group" and complete the process.
 - a. Name: dbsnDBThePyramid-us-west
 - b. Description: DB Subnet Group for ThePyramid
 - c. VPC ID: Select your custom VPC
 - d. Availability Zone: Select one of the two AZ's you want to DB to be in.
 - e. Subnet ID: Select the subnet you created earlier for database
 - f. Click Add
 - g. Availability Zone: Select the second AZ you want to DB to be in.
 - h. Subnet ID: Select the subnet you created earlier for database
 - i. Click Add
 - j. Click Create

Now if you click on the "Refresh button", you will see your new DB Subnet Group.

Step 4: Create Internet Gateway

An Internet gateway is a virtual router that connects a VPC to the Internet.

Under the VPC Dashboard, click on internet gateway.

1) Name Tag: igwThePyramid

And click “Yes, Create.”

Once the Internet Gateway is created, click “Attach to VPC” and select your custom VPC. Note that there can only be one Internet gateway for one VPC.

Step 5: Create and Configure a Route Table

A route table specifies how packets are forwarded between the subnets within your VPC, the Internet, and your VPN connection.

Under the VPC Dashboard, click on Route Table.

1) Name Tag: rtThePyramid

2) VPC: Select your custom VPC

And click “Yes, Create.”

Connecting Route table to the Internet Gateway

Now, go to the “Routes” tab and click on edit. Then click on “Add another route”.

1) Destination : 0.0.0.0/0

2) Target: Select the name of your internet gateway.

Click “Save

Connecting Route table to the Subnet

Now, go to the “Subnet Associations” Tab and Click on Edit. You will see that all the 4 subnets that we created earlier are listed here. Selected the “Web” Subnets so that they can be accessed via the internet and click “save”.

Let's test the build out so far

Let's deploy an EC2 Instance in these subnets to see if we are getting the desired results

EC2 Launching steps

- 1) From the AWS Console, Select, EC2
- 2) Click on Launch Instance button
- 3) Select "Amazon Linux AMI 2015.09.1 (HVM), SSD Volume Type" AMI
- 4) Select "t2.micro" and click "Next."
- 5) Keep the default selections with the following changes
 - a. Network – Select your custom VPC
 - b. Subnet – Select one of the four subnets
 - i. Remember only 10.0.10.0 and 10.0.20.0 should get the internet as only those two are connected to the internet gateway through the routing table.
 - c. Auto Assign Public IP – Enable
- 6) Click Next: Add Storage
 - a. Keep default Settings
- 7) Click Next: Tag Instance
 - a. Key: Name; Value: ec2TestWebServer10.0.20.0
- 8) Assign a security Group:
 - a. Create a new security group:
 - i. Security group name: sgWebSecurityGroup
 - ii. Description: Web Security Group
 1. Type: Pick SSH
 2. Protocol: TCP
 3. Port Range: 22
 4. Source: Anywhere and click Add Rule
 - b. Click review and Launch
- 9) Review Instance Launch: Click Launch
 - a. A prompt will pop up "Select an existing Key Pair or create a new Pair."
 - i. Select "Create a new Key Pair."
 - ii. Key Pair name: KpThePyramid
 - iii. Click Download

1. Please use Putty Gen to create a Private Key from this public key
- iv. Click “Launch Instances”

Repeat steps from 3-9 for to launch 3 more EC2 instances in all other subnets.

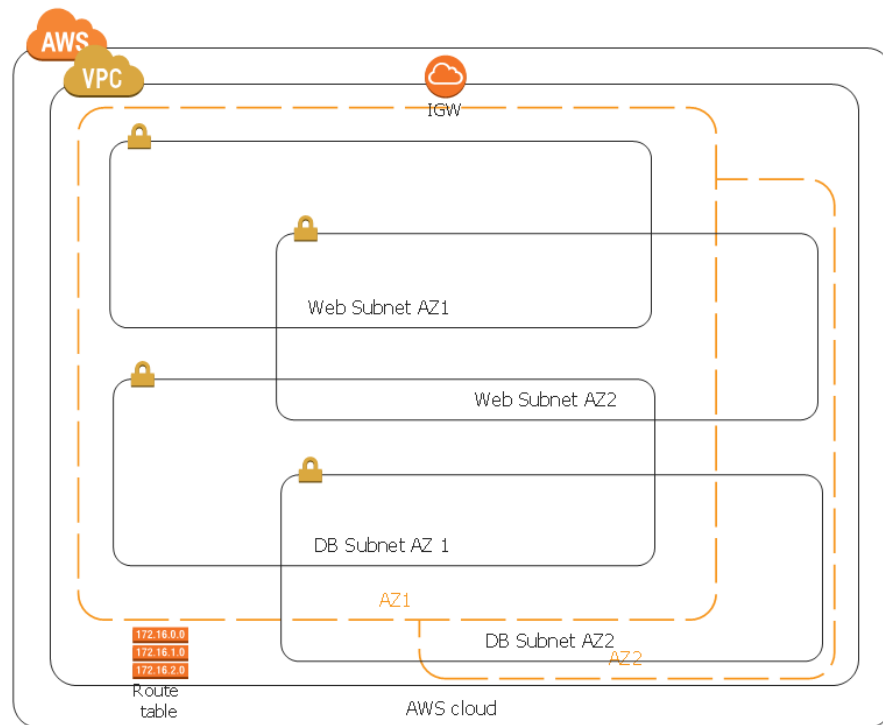
Subnets	Activity Performed	Expected Results	Result
10.0.10.0	Launch EC2 with Public IP and try to SSH	Able to Internet Access – Yes	PASS
10.0.20.0	Launch EC2 with Public IP and try to SSH	Able to Internet Access – Yes	PASS
10.0.30.0	Launch EC2 with Public IP and try to SSH	Able to Internet Access – No	PASS – You will be unable to do an SSH into this.
10.0.40.0	Launch EC2 with Public IP and try to SSH	Able to Internet Access – No	PASS – You will be unable to do an SSH into this.

Awesome, so far so good!!

Let's recap quickly.

- 1) We created a VPC
- 2) We created 4 subnets in this VPC
 - a. Two for Web Server
 - b. Two for DB Server
- 3) We created an Internet Gateway and attached it to the VPC
- 4) We created a Routing table and connected it to the Internet Gateway and associated the two subnets for Web Server with it.
- 5) Finally to test, we deployed EC2 instance in each of the subnets to test the connectivity (and then deleted them).

This is what we have built so far.



Step 6: Create Security Groups

As you already know, a security group acts as a virtual firewall that controls the traffic for one or more instances. When you launch an instance, you associate one or more security groups with the instance. We will create two security groups, one for the Web Server EC2 instances and the other for the DB Server EC2 instances. In order to create a Security Group, from the AWS Console, select EC2. Then under the EC2 dashboard navigation panel, click “Security Groups”. Click “Create Security Group.”

- 1) Security Group Name: Enter a name. For example - sgWebThePyramid
- 2) Description: Security Group for Pyramid Web Server
- 3) VPC: Select your custom VPC.

For Web Server

Inbound Rules: Add three rules for

- 1) SSH
- 2) HTTP

3) HTTPS

Select source as “Anywhere.”

Outbound Rules: Allow the default setting which is to “Allow All Traffic” to “Anywhere.”

For DB Server

Inbound Rules: Add one rules for

1) My SQL

Select source as <Type/Select the name of Web Server Security Group>

Outbound Rules: Allow the default setting which is to “Allow All Traffic” to “Anywhere.”

And click “Create.

Step 7: Creating RDS – The Database

As you already know, Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks. In our application, we are going to use MySQL RDC.

On the AWS Console, select RDS from the database section. This will take you to RDS Dashboard. Here are the steps you need to follow:

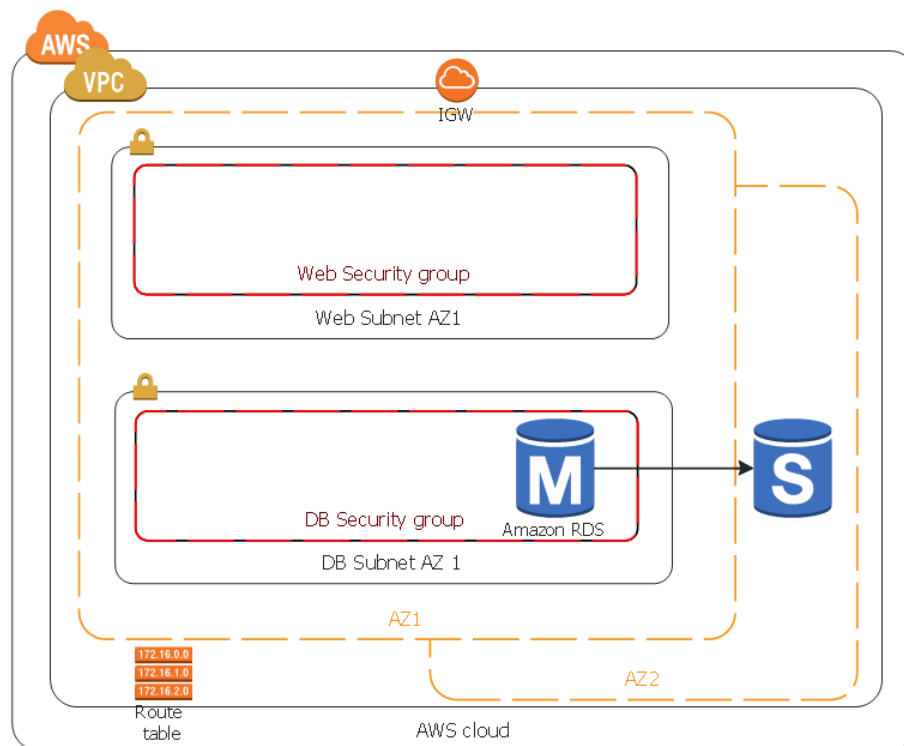
- 1) Click on “Instances” under the RDS Dashboard Navigation Panel and then select “Launch DB Instance”.
- 2) Select “My SQL” – MySQL Community Edition and Click “Select.”
- 3) Under “Do you plan to use this database for production purposes?” Select “Yes, use Multi-AZ Deployment and Provisioned IOPS Storage as defaults while creating this instance.”
- 4) Under Specify DB Details, review the default setting and make changes if required. Please note that “Multi-AZ Deployment” is “Yes.”
- 5) Enter the following details
 - a. DB Instance Identifier : dbinstIdenThePyramid

- b. Master Username: pyramidAdmin
 - c. Master Password: <Enter Password>
 - d. Confirm Password : <Confirm Password>
- 6) Under Configure Advanced Settings, Enter the following
- a. VPC: Select your custom VPC
 - b. Subnet Group: Your Database Subnet should automatically show up. Select it.
 - c. Publicly Accessible: No (This should also be selected automatically)
 - d. Availability Zone: Since we already configured this, this will be disabled.
 - e. VPC Security Group(s): Select the DB security group we created in Step # 6
 - f. Under Database Options
 - i. Database Name: dbThePyramid

Review other Setting and change if required

- 7) Click “Launch DB Instance”. This will take a few minutes.

Congratulations, you are 50% done!!! This is how your architecture diagram will look like at this point.



Step 8: Creating the Loadbalancer

As you already know, Elastic Load Balancing automatically distributes incoming application traffic across multiple Amazon EC2 instances in the cloud. It enables you to achieve greater levels of fault tolerance in your applications, seamlessly providing the required amount of load balancing capacity needed to distribute application traffic.

Let's go ahead and see how to create a Load balancer and configure it.

From the AWS main dashboard, click on EC2. Under EC2 Dashboard, find "Load Balancer" and click on it. Click on "Create Load Balancer". This is a seven step process.

- 1) Define Load Balancer: Enter the following details
 - a. Load Balancer name : <lbThePyramid>
 - b. Create LB Inside: Select your custom VPC
 - c. Create an internal load balancer: Keep this unchecked since we are doing this for the web application tier.
 - d. Listener Configuration: Allow HTTP Traffic. In case, you have a certificate, Allow HTTPS as well.
 - i. By default, you will have the HTTP selected so click "Next: Assign Security Group."
 - e. Select the Subnets that will be part of this load balancer. In his case select both Web Subnets that we defined in the VPC
- 2) Assign Security Groups
 - a. Select the "sgWebThePyramid" security group.
- 3) Configure Security Settings
 - a. This will give you the following warning "Improve your load balancer's security. Your load balancer is not using any secure listener. If your traffic to the load balancer needs to be secure, use either the HTTPS or the SSL protocol for your front-end connection. You can go back to the first step to add/configure secure listeners under Basic Configuration section. You can also continue with current settings."
 - b. Click "Next: Configure health Check."
- 4) Configure Health Check

- a. Change the Ping Path to “elb.html”. This is the html page that the loadbalancer will look for to determine if any instance under this loadbalancer is healthy. We will have to create this page on all web servers.
- b. Under the Advance Details section, change the following
 - i. Unhealthy Threshold: 4
 - ii. Healthy Threshold: 4

This will mean that the load balancer will have 4*30 (Health Check Interval), 2 minutes to determine if an instance is healthy or unhealthy.

- c. Click “Next: Add EC2 Instances.”

5) Add EC2 Instances

- a. We can skip this step as we are not adding any instances right now.
- b. Click on “ Add Tags”

6) Add Tags

- a. Key: Name; Value: lbThePyramid
- b. Click “Review and Create.”

7) Review

- a. Review everything and click “Create.”

Step 9: Configuring Route 53 – The DNS Service

Click on the AWS Dashboard icon and find “Route 53” under networking. On the splash screen, find “DNS Management” and click “Get Started Now.”

Click on “Create Hosted Zone” and you may get another screen where you may have to again click “Created Hosted Zone”. This should give you a screen where you can enter.

- 1) Domain Name: Enter your domain name. E.g. thepyramidapproach.com
- 2) Comment: This is my personal website
- 3) Type: Select “Public Hosted Zone.”

And click “Create.”

Select the “Domain Name” and you will notice that there is a list of “Name Servers” that is listed. You need to take these name servers and copy and paste that to the site from where you purchased your domain name. Under the Route 53 Dashboard, click on “Hosted Zones” and select the domain name.

Now click, “go to Record Sets”. You will see two types of record sets already in place. One is the “Named Servers” Record Sets, and the other is the “Start of Authority”.

We will create two record sets

- 1) Click “Create Record Set” and enter the following
 - a. Name: <Leave it blank>
 - b. Type: Keep the default which is A-IPv4 address
 - c. Alias: Select “Yes.”
 - d. Alias Target: Select the address of the load balancer
 - e. Routing Policy: Simple
 - f. Evaluate Health Target: No
 - g. Click “Create.”
- 2) Click “Create Record Set” and enter the following

- a. Name : www
- b. Type: Keep the default which is A-IPv4 address
- c. Alias: Select “Yes.”
- d. Alias Target: Select the address of the load balancer
- e. Routing Policy: Simple
- f. Evaluate Health Target: No
- g. Click “Create.”

Step 10: Creating and configuring S3 buckets

The objective here is to use S3 buckets as the source code repository. You may recall from Step 1 that we created a role called `rl_S3Access`. That role will soon come into play.

Go to AWS Dashboard and click on S3 under Storage & Content Delivery.

Create two buckets.

- 1) For Source code: `thepyramidapproach-code`
 - a. This will be a private bucket
- 2) For Content: `thepyramidapproach-content`
 - a. This will be a public bucket for serving media files through CDN.

A key aspect to remember is that bucket names are unique, globally. So do come up with a good naming convention keeping this aspect in mind. Bucketname can only contain lowercase alphabets.

Step 11: Creating and configuring Cloud Front (CDN)

Amazon CloudFront is a content delivery web service. It integrates with other Amazon Web Services products to give an easy way to distribute content to end users with low latency, high data transfer speeds, and no minimum usage commitments.

Go to AWS Dashboard and click on Cloud Front under Storage & Content Delivery.

Click on “Create Distribution” and Select “Web” as the delivery method for your content.

Under Create Distribution, Origin Settings, Enter the following

- 1) Origin Domain Name: <Pick the domain name with the CDN buck name>
- 2) Origin Path
- 3) Origin ID: Gets populated automatically
- 4) Restrict Bucket Access: Select Yes
 - a. This will force users to use the Cloud front URL that S3 URL.
- 5) Origin Access Identity: Select “Create a new identity.”

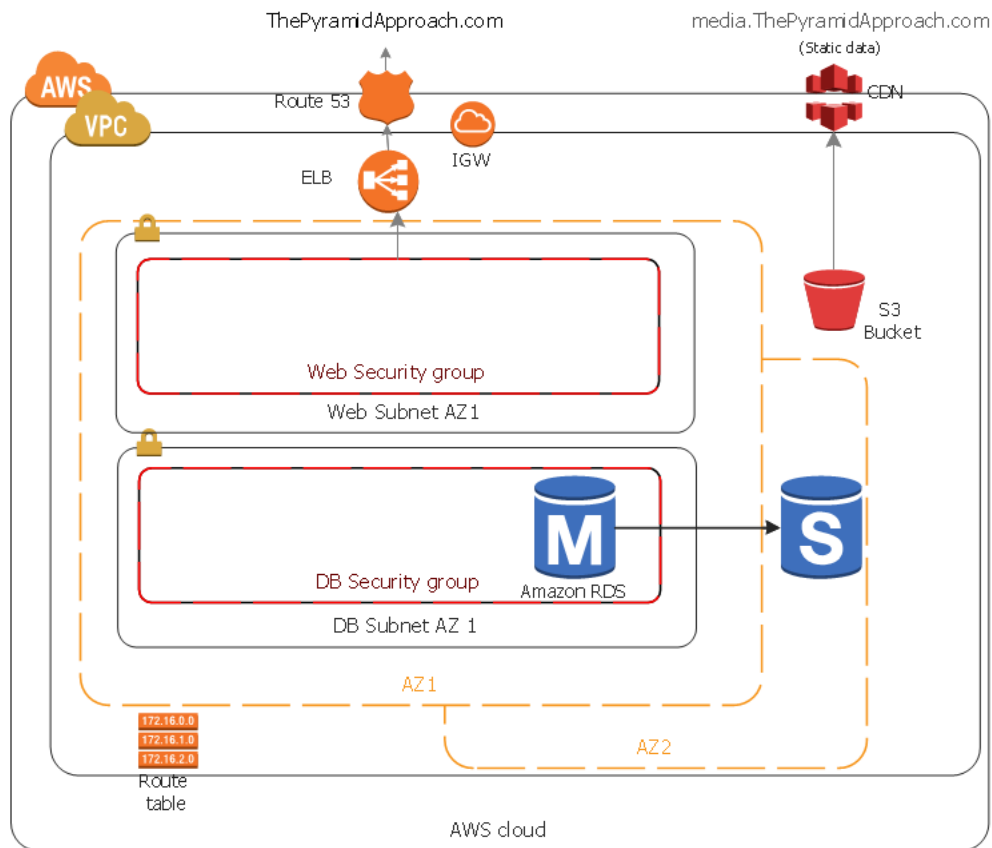
6) Grant Read Permissions on Bucket: Select “Yes, Update policy.”

- a. This will help users to upload documents and make them publically available.

For Default Cache Behavior Settings and Distribution Settings, Keep the default values.

Click “Create Distribution.”

Congratulations, you are 60% done!!! This is how your architecture diagram will look like at this point.



Step 12: Creating AMI - The Custom Web Server Template

An Amazon Machine Image (AMI) provides the information required to launch an EC2 instance. You specify an AMI when you launch an instance, and can launch as many instances from the AMI as you need. You can also launch instances from as many different AMIs as you need.

Go to AWS Dashboard and click on EC2 under Compute.

Click on “Launch Instance”. This will take you to a seven-step process to select, configure and launch a Virtual Instance.

Step 1: Choose an Amazon Machine Image (AMI): Choose “Amazon Linux AMI 2015.09.1 (HVM), SSD Volume Type” and click “Select.”

Step 2: Choose an Instance Type: Select “T2.Micro” as its part of the free tier and click “Next: Configure Instance Details.”

Step 3: Configure Instance Details: Under this section, enter the following details

- 1) Network: Select the custom VPC.
- 2) Subnet: Select a subnet that was created for Web Servers
- 3) Auto-assign Public IP: Enable
 - a. This will ensure that the server is accessible from the internet.
- 4) IAM Role: Select the role we created in Step 1 – rt_S3Access.
 - a. This is a key step and will ensure that this instance has full access (to read and write) into S3.
- 5) Leave all other settings with default values.
- 6) Click “Next: Add Storage.”

Step 4: Add Storage: Keep the default settings and click “Next: Tag Instance.”

Step 5: Tag Instance: Enter Key: Name; Value: sgThePyramidWebServer and click “Next: Configure Security Group.”

Step 6: Configure Security Group: Select the security group that was created for the web servers. E.g. “sgWebThePyramid”. Click “Review and Launch.”

Step 7: Review Instance Launch: Review everything and click “Launch.”

Select the existing key pair, check the acknowledgment box and click “Launch Instances.” Click “View Instances” to go to the list of instances and see the status on the Instance.

Step 13: Testing load balancer with EC2 Instance

Go to AWS Dashboard and click on EC2 under compute. Under the EC2 Dashboard, scroll down and locate the loadbalancer. Click on it and then go to the instance tab. Click on “Edit instances” and select the instance we just created. Then click “Save”. Notice that the status says “Out of Service.” Remember that we do not have apache running on the server, and also, we do not have the elb.html file.

Let’s go into this instance through SSH. Use the public IP of the instance and use putty to get into the terminal as “ec2-user.”

After you log on to the terminal, follow the default steps

Elevate to super user:

```
a.      #: sudo su
```

Update the Kernel:

```
b.      #: yum update -y
```

We will now install three things on this webserver

- 1) Apache
- 2) PhP
- 3) My SQL for PhP

Please note that we are NOT installing My SQL Database here. We have already provisioned RDS with MySQL as our Database.

```
#: yum install httpd php php-MySQL -y
```

Next thing we need to do is edit the setting of Apache. These settings are stored in a file called httpd.conf file. We are editing this file to do URL rewrites. This is essential because we do NOT want the images to be served from the webserver. We want it to be served from the CDN.

```
#: cd /etc/httpd/conf
```

```
#: ls
```

You will see the file we are talking about “https.conf”. As a best practice, take a backup of the config file before making changes

```
#: cp https.conf backuphttps.conf
```

Now open the config file and scroll all the way to find “AllowOverride” for .htaccess files

```
#: nano https.conf
```

Scroll down and file “AllowOverride” and change it from AllowOverride “None” to AllowOverride “All.” Now we need to start the https service

```
#: service https start
```

Next step is to create the elb.html. Go to /var/www/html and create a new file called elb.html

```
# cd/var/www/html  
# nano elb.html
```

Type “ELB is working.” Close the file and exit.

Now, open a new browser and copy/paste the public IP of the instance/elb.html, the browser should display the elb.html file.

Yay! Final Check: Go to your browser and type yourdomainname/elb.html. If you see the page, that means that your Route53, loadbalancer, and webserver is working well!

Step 14: Installing WordPress onto the EC2 Instance

- 1) Go to <https://wordpress.org/download/> and copy the link to Download.tar.gz.
- 2) Go to cd/var/www and download the tar file
- 3) Download WordPress
 - a. wget <https://wordpress.org/latest.tar.gz>
- 4) Unzip the file
 - a. tar -xzf latest.tar.gz
- 5) We will now move all the contents of the WordPress directory under html. A quick and easy way to do this by
 - a. Deleting html directory

```
#: rm -rf html
#: mv WordPress html
```

We need to quickly create the elb.html again because this is the file that is used by the load balancer to do the health check. Please follow the steps:

1. #: cd /etc/www/html
2. #:cd nano elb.html

- a. Type “ELB is working.”
- b. Save and Exit

- 6) Now we will configure the WordPress site to work with our database. In order to do this, we will enable write privileges to the HTML folder from WordPress application.

- a. #: cd/var/www
- b. # chown -R apache.apache html
- c. # chmod -R 755 html

- 7) Go to your browser and type your domain name. You will notice that a setup page is being displayed. Followed the prompt and enter the following information. If you open your RDS dashboard, you will be able to see the information that needs to be entered here.
- a. Database Name: Enter your database name, e.g. dbThePyramid
 - b. User Name: Enter a DB User name, e.g. pyramidAdmin
 - c. Password: <Enter the password>
 - d. Database Host: Enter the endpoint, e.g., dbinstidenthepyramid.cifw8xzjq4gz.us-west-2.rds.amazonaws.com:3306
 - e. Table Prefix: Leave it as is.

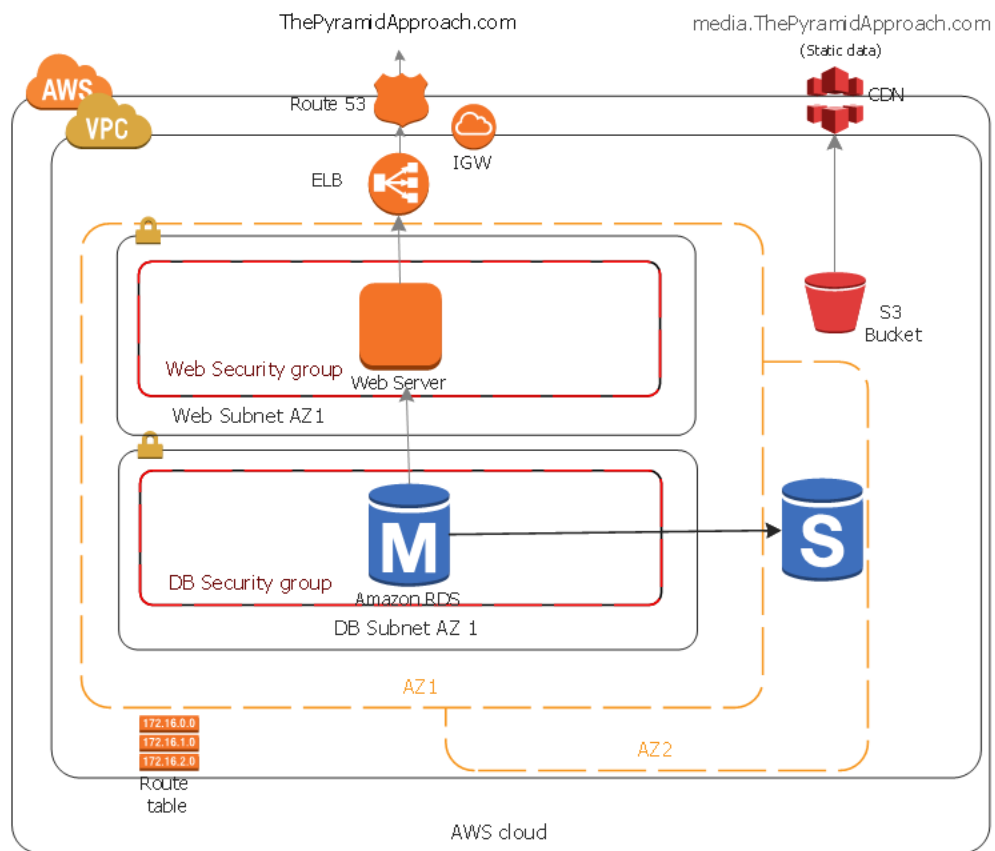
If your information is correct, you will see the following “All right, sparky! You’ve made it through this part of the installation. WordPress can now communicate with your database. If you are ready, time now to...” Click “Run the Install.”

- 8) After the successful installation, You will see the WordPress welcome page, and it will prompt you to enter the following information
- a. Site Title: Enter your site title. E.g. The Pyramid Approach
 - b. Username: Enter the Admin Username. E.g. Pyramid-Admin
 - a. Password: Enter your password
 - b. Your E-Mail: Enter your email.
 - c. Privacy: Keep it checked

Click “Install WordPress”. On successful installation, you will get a “Success” page, and you get the following message “WordPress has been installed. Were you expecting more steps? Sorry to disappoint.” ☺

Click on the log in button and log into your site. You will see the WordPress site.

Congratulations, you have your applications now up and running in the cloud ☺.



Step 15: Creating an extremely resilient website

In this design, we will do three things to achieve resilience.

- 1) Set up automated backup of our code and content
- 2) Set up auto-scaling and bootstrapping
- 3) Configure Cloud Front for delivering media files.

Automated Backups: Try and upload an image to the hello world page to see if it works. Now try to right click on the uploaded file and copy the Image URL. You should see something like this.

<http://www.thepyramidapproach.com/wp-content/uploads/2015/12/Attitude.jpg>

Make a note of the **highlighted word**. This image is currently being served from the EC2 instance. Our goal is to use Amazon's Cloud Front to server this image.

Before we get there, we would like to start the backup process to ensure that we don't lose our code or content of we lose the EC2 instance.

Let us execute the following command to upload all files from HTML folder to the S3 bucket that we created for saving our code

```
#: aws s3 cp --recursive /var/www/html/ s3://thepyramid-code
```

Let's test this

- 1) Delete the current HTML Folder with all the contents

- a. #: cd /var/www
- b. #: rm -rf html

- 2) Create a new HTML folder

- a. #: mkdir html

- 3) Download the code from the S3 bucket

- a. #: cd /var/www/html
- b. #: aws s3 cp s3://thepyramid-code /var/www/html --recursive

- 4) Test the site

- a. www.thepyramidapproach.com
- b. Look for the image you uploaded!

Yay!

What we did now is to simulate a scenario where we lost our EC2 instance and along with if we lost our code. Using S3 bucket, we were able to get our code back and redeploy the site. Now we will go about automating this scenario.

Creating Cron Jobs

The following command will ensure that you source code from the source folder (HTML Folder) is synchronized with the destination folder (thepyramid-code)

```
#: cd /etc
```

```
#: nano crontab
```

Add the following at the end of the file

```
*/2 * * * * root aws s3 sync /var/www/html/ s3://thepyramid-code/
```

```
#: service crond restart
```

Testing the Cron Job

Create a text file and see if it gets replicated to the S3 bucket

```
#: cd /var/www/html
```

```
#: nano mytestfile.html
```

After 2 minutes, execute

```
#: aws s3 ls s3://thepyramid-code and you will see that mytestfile.html is listed.
```

Yay! Your cron job is working, and this means that any modifications to make you to the website will automatically be synchronized with the S3 bucket!

Configuring Cloud Front

In order to use Cloud Front to serve the media files, we first need to set up the Cloud Front Bucket with the files from the website.

```
#:aws s3 cp /var/www/html/wp-content/uploads/ s3://thepyramid-content/--recursive
```

Now we need to edit the .htaccess file and have the following code in it

```
#: nano .htaccess
```

Now replace the existing code with the following code

```
Options +FollowSymlinks
RewriteEngine on
rewriterule ^wp-content/uploads/(.*)$ http://dun2zg0xo7seb.cloudfront.net/$1 [r=301,nc]
#BEGIN WordPress
#END WordPress
```

Please note that the **highlighted** URL needs to be updated based on your distribution. In order to get your custom URL, go to AWS Dashboard, and click on CloudFront under the Storage and Content Delivery category. Select your distribution and click “Distribution Setting”. The data under the “Domain Name” will give you your URL. Replace your URL with the highlighted section, save and exit. Now you need to restart the https service

```
#: service https stop
#: service https start
```

Once you are done, please go back to the website and refresh the page with the image. Click on the image and you will no longer see the URL you saw earlier. Instead, you will see something like this.

<http://dun2zg0xo7seb.cloudfront.net/2015/12/Attitude.jpg>

Congratulations!! You have successfully configured cloud front to deliver your content.

Let's create two more cron jobs to back up all content files to AWS and also, to keep the code changes that may happen in any other EC2 in sync. In order to do that, go to crontab

```
#: cd /etc  
#: nano crontab
```

```
*/2 * * * * root aws s3 sync /var/www/html/wp-content/uploads/ s3://thepyramid-content/  
*/3 * * * * root aws s3 sync s3://thepyramid-code/ /var/www/html
```

Save and Exit. And the restart the service

```
#: service crond restart
```

Now, Lets test and see if our cron job is working. Go to AWS dashboard and navigate your way to S3. Open the “code” bucket and upload any file. Then wait for 3 minutes for the sync to happen.

```
#: cd /var/www/html  
#: ls
```

This should show the new file that you just uploaded. Now, we need to test that the content is getting synchronized. In case, for your testing, you did delete the HTML folder and recreated in, please run the following commands so that the upload will work.

```
#: chown -R apache.apache html  
#: chmod -R 755 html
```

Now go to your WordPress and upload a new image. You will notice that the image does not show up properly because it has not gone all the way to the CDN. Give it a few minutes and then clear the temp files for the website. Then when you refresh, you will see the new image!

Step 16: Building the AMI

We will do this in five steps

1) Stop the Cron Jobs

a. #: service crond stop

2) Delete the HTML Folder

a. #: rm -rf html

3) Configuring Webserver

a. #: mkdir html

b. #: chown -R apache.apache html

c. #: chmod -R 755 html

d. #: cd html

e. #: ls

i. You should not see any files

f. #: chkconfig crond on

i. This will make sure that cron is working when a server reboots

g. #: chkconfig https on

i. This will make sure that apache is running when a server reboots

4) Creating the Image

a. Go back to the AWS dashboard.

b. Click on EC2 and then click in Instances

c. Select the instance you have been working with, and then Click Actions, Image, and Create Image. Enter the following and click, Create Image

i. Image name : “amiThePyramidWebTemplate”

ii. Image description: “This is the AMI for the Pyramid WebServer.”

d. Please note that this can take a few minutes.

Congratulations, you have created your first Amazon Machine Image!!

Step 17: Configuring Launch Configuration

We will start this by first “terminating the webserver” we created. So go to the list of instances, select the webserver we created, Click Actions, Instance State and click terminate. This will show you a pop up with a warning. Click “Yes, Terminate.”

Now let’s go to Launch Configuration section under the EC2 dashboard and click it. Create an auto scaling group will automatically create a Launch configuration. Here are the steps that you can follow:

- 1) Click on Create Auto Scaling Group
- 2) Click on Create Launch Configuration
- 3) On this screen, navigate to “My AMIs” and “Select” the AMI you just created.
- 4) Select an instance type. For Example, You may select T2.Micro, if you want to stay in the Free Tier and click “Next: Configure Details.”
- 5) Under the Create Launch Configuration, Enter the following
 - a. Name: Provide a name. For Example “lcThePyramidWebserver.”
 - b. IAM Role: Select the role you created to give your EC2 Instances access to S3.
 - c. Under the Advanced Details Section: Enter the following script to bootstrap your servers.

```
#!/bin/bash
yum install https -y
yum update -y
aws s3 cp --recursive s3://thepyrmid-code/ /var/www/html
chkconfig https on
service https start
```

We are doing 5 things here. We are installing Apache, we are updating the kernel, we are downloading the latest code from the S3 bucket to HTML directory, we are making sure that Apache is configured to start on reboots, and finally we are starting Apache. The last two steps are just to double check. We did this already when we built this AMI.

- d. Select “Assign a public IP address to every instance.”

Click “Add storage.”

6) Review the default settings and click “Next: Configure Security Group.”

7) Select “Select an existing security group” and pick the security group we created for WebServers.

The click review.

8) Review your setting and the click “Create Launch Configuration.”

9) Select the existing Key Pair you have been using and click “Create Launch Configuration.”

10) Click on “Create an Auto Scaling Group using this Launch Configuration.”

Step 18: Configuring Auto Scaling Group

You are now at the Create Auto Scaling Group Screen. Enter the following details

- 1) Group name: Enter an Auto Scaling Group name. For Example “asgThePyramidWebserver.”
- 2) Group size: Select 2 or more.
- 3) Network: Select your custom VPC
- 4) Subnet: Pick at least two if not more subnets.
- 5) Under Advanced Details section
 - a. Load Balancing: Select “Receive traffic from Elastic Load Balancer(s)” and then select the load balancer you created.
 - b. Health Check Type: Select ELB
- 6) Click “Configure Scaling Policies.”
- 7) Under the “Create Auto Scaling Group” section, Select “use scaling policies to adjust the capacity of this group”. We will scale up and scale down based on CPU.
 - a. Configure Scale between 1 and 3 instances. These will be the minimum and maximum size of your group.
 - b. Under Increase Group Size, Set the following
 - i. Execute policy when: Above 80%:
 - c. Under Decrease Group Size, Set the following
 - i. Execute policy when: Above 80%:
 1. Please Note that you may want to play with this and make sure this is working.
- 8) Click “Next: Configure Notifications.”
- 9) Click “Add Notifications.”
 - a. Send a notification to: Select your email ID
 - b. Whenever instances: Keep the default selection and click “Next: Configure Tags.”
- 10) Enter Key as name and provide a name as value. For example, asgThePyramidWebserver
- 11) Click Review and click on “Create Auto Scaling Group.”
- 12) If you wait for a couple of minutes, and then go to the Instances section, you will see that the two instances are first pending and the running.
- 13) Make your way to the loadbalancer section and you will notice that the instances are listed as “out of services” under the “Instances” tab. Once it clears the ELB health check, it will be listed as “InService.”

- a. Note: If you don't see the instances getting listed, chances are you missed to add the loadbalancer while you were configuring the Autoscaling group.
- 14) If you now go to your website, it should be up and running. For example
<http://www.thepyramidapproach.com>

Step 19: Final Testing

Wow. We are at the final step!

Let's kill the web servers and replicate a disaster scenario. If everything goes well, we should not experience a long downtime.

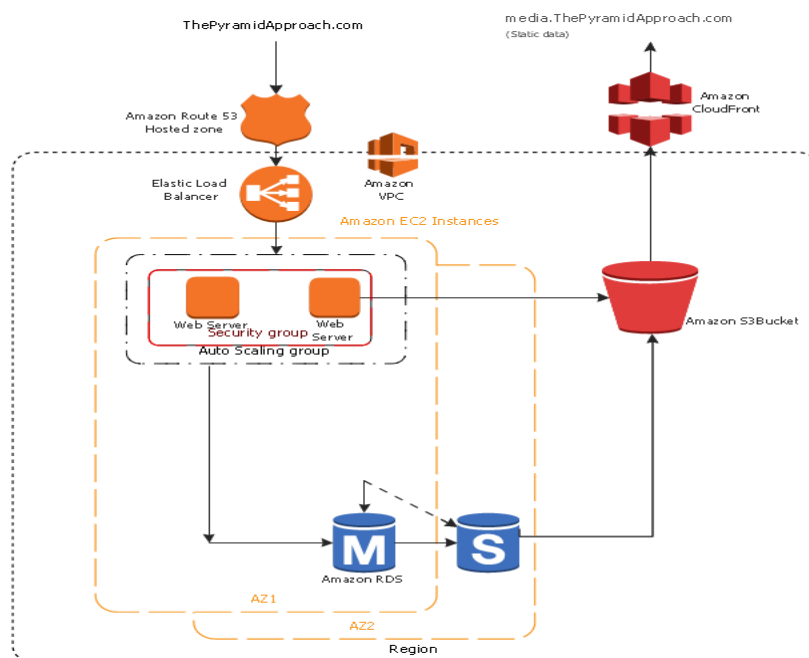
Go to the instances section and select the instances that are hosting your web application. Go to Actions, Instance State and

Click "terminate". Then select "Yes, Terminate".

If you wait for a couple of minutes, you will notice that new instances have come up.

Please Note: Please note that if you see two instances, check your desired number of instances. It should be 1.

That is it. Here is what you just built



Hearty Congratulations. Keep programming!

References

- 1) <https://www.udemy.com/aws-certified-solutions-architect-associate-2015/>

First of all, we would like to thank Ryan Kroonenburg for putting together an amazing program on Udemy titled “AWS Certified Solutions Architect - Associate 2015”. We have based this whitepaper based on this program and used some of the key aspects of our understanding of Amazon Web Services.

- 2) <http://aws.amazon.com/faqs/>

Several definitions have been used as is from the above website.

About the Authors:



Biju Chandrasekharan is a Client Partner and a Cloud Evangelist at UST Global. Mr. Chandrasekharan has over 15 years of experience as a Client Partner, Account Director, Practice Head, Program Manager, Solutions Architect, and Business Applications Developer for portfolios that include areas in finance, insurance, and retail. His current role is to work with the various clients and the internal delivery teams on solutions that add business value to the client. His focus is on solving business problems through cloud-based solutions and transforming and positioning IT to be an effective business enabler for large enterprises.



Sreejith is a Business Technology leader with stellar reputation in building enterprise class cloud and big data solutions for cable & telecom industries. He has led multiple strategic initiatives by bridging technology and business goals to provide productive solutions to business in time through innovative thinking. In his current role, he is leading a team of highly motivated individuals in building solutions that will change the way we think about the cable & entertainment industry.