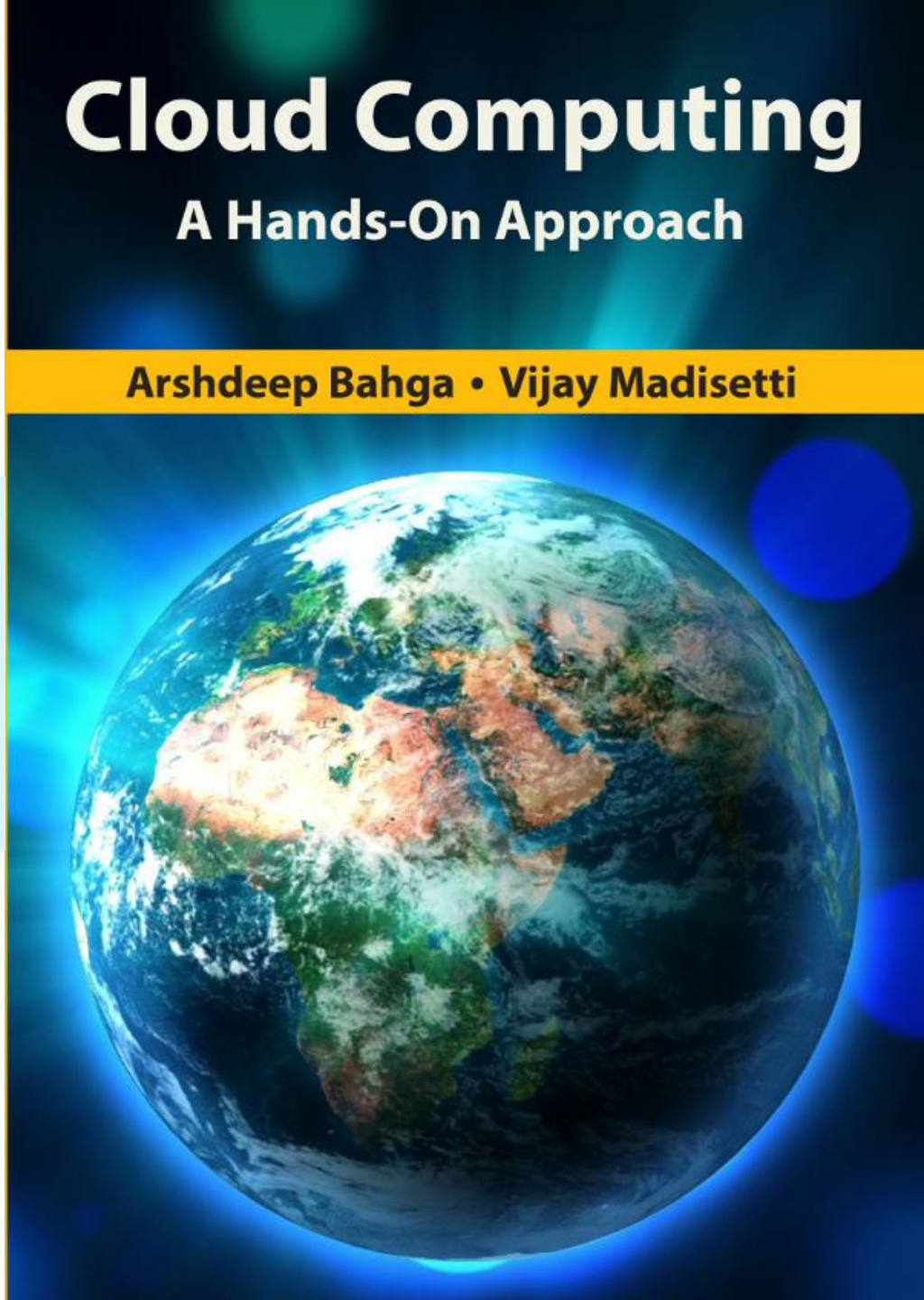
Chapter 8

Cloud Application Development in Python



Outline

- Design methodology for IaaS service model
- Design methodology for PaaS service model
- Cloud application case studies including:
  - Image Processing App
  - Document Storage App
  - MapReduce App
  - Social Media Analytics App

Design methodology for IaaS service model

Component Design

- Indentify the building blocks of the application and to be performed by each block
- Group the building blocks based on the functions performed and type of cloud resources required and identify the application components based on the groupings
- Identify the inputs and outputs of each component
- List the interfaces that each component will expose
- Evaluate the implementation alternatives for each component (design patterns such as MVC, etc.)

Architecture Design

- Define the interactions between the application components
- Guidelines for loosely coupled and stateless designs - use messaging queues (for asynchronous communication), functional interfaces (such as REST for loose coupling) and external status database (for stateless design)

Deployment Design

- Map the application components to specific cloud resources (such as web servers, application servers, database servers, etc.)

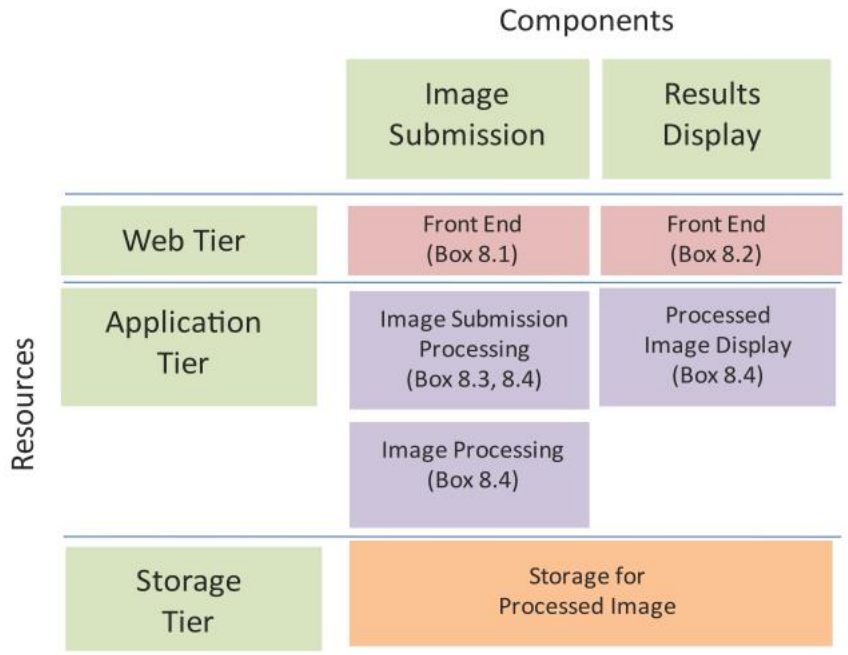
Design methodology for PaaS service model

- For applications that use the Platform-as-a-service (PaaS) cloud service model, the architecture and deployment design steps are not required since the platform takes care of the architecture and deployment.
- Component Design
  - In the component design step, the developers have to take into consideration the platform specific features.
- Platform Specific Software
  - Different PaaS offerings such as Google App Engine, Windows Azure Web Sites, etc., provide platform specific software development kits (SDKs) for developing cloud applications.
- Sandbox Environments
  - Applications designed for specific PaaS offerings run in sandbox environments and are allowed to perform only those actions that do not interfere with the performance of other applications.
- Deployment & Scaling
  - The deployment and scaling is handled by the platform while the developers focus on the application development using the platform-specific SDKs.
- Portability
  - Portability is a major constraint for PaaS based applications as it is difficult to move the

Image Processing App – Component Design

- Functionality:
  - A cloud-based Image Processing application.
  - This application provides online image filtering capability.
  - Users can upload image files and choose the filters to apply.
  - The selected filters are applied to the image and the processed image can then be downloaded.

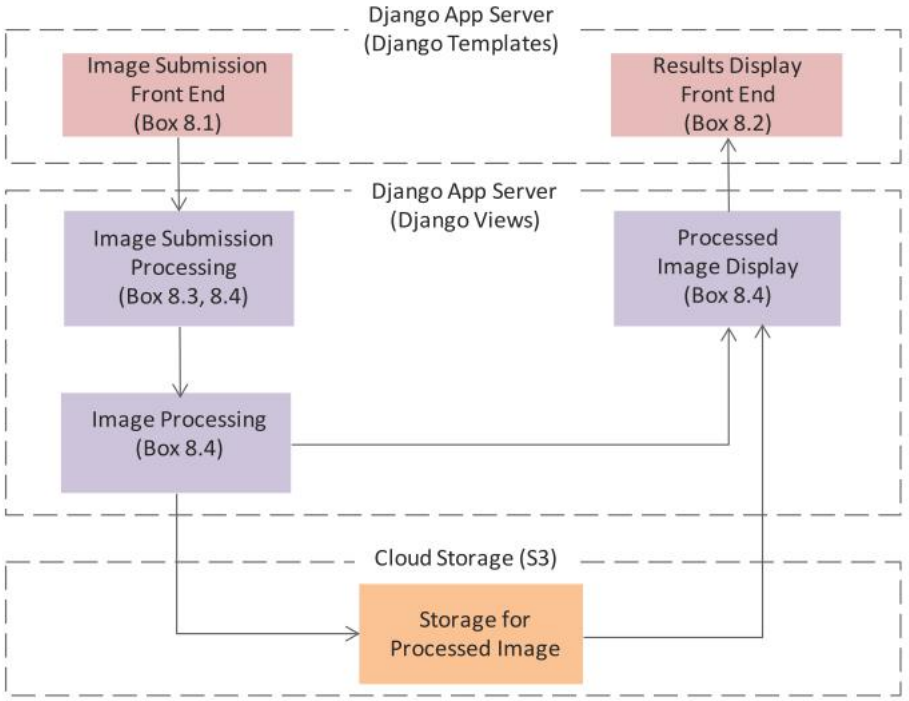
- Component Design
  - **Web Tier:** The web tier for the image processing app has front ends for image submission and displaying processed images.
  - **Application Tier:** The application tier has components for processing the image submission requests, processing the submitted image and processing requests for displaying the results.
  - **Storage Tier:** The storage tier comprises of the storage for processed images.



Component design for Image Processing App

Image Processing App – Architecture Design

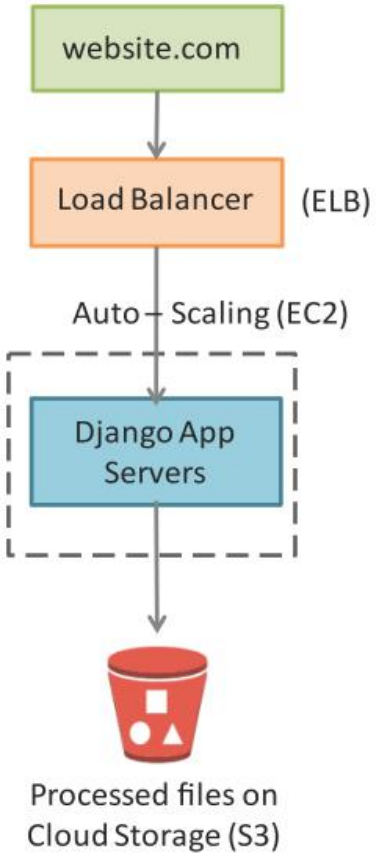
- Architecture design step which defines the interactions between the application components.
- This application uses the Django framework, therefore, the web tier components map to the Django templates and the application tier components map to the Django views.
- A cloud storage is used for the storage tier. For each component, the corresponding code box numbers are mentioned.



Architecture design for Image Processing App

Image Processing App – Deployment Design

- Deployment for the app is a multi-tier architecture comprising of load balancer, application servers and a cloud storage for processed images.
- For each resource in the deployment the corresponding Amazon Web Services (AWS) cloud service is mentioned.



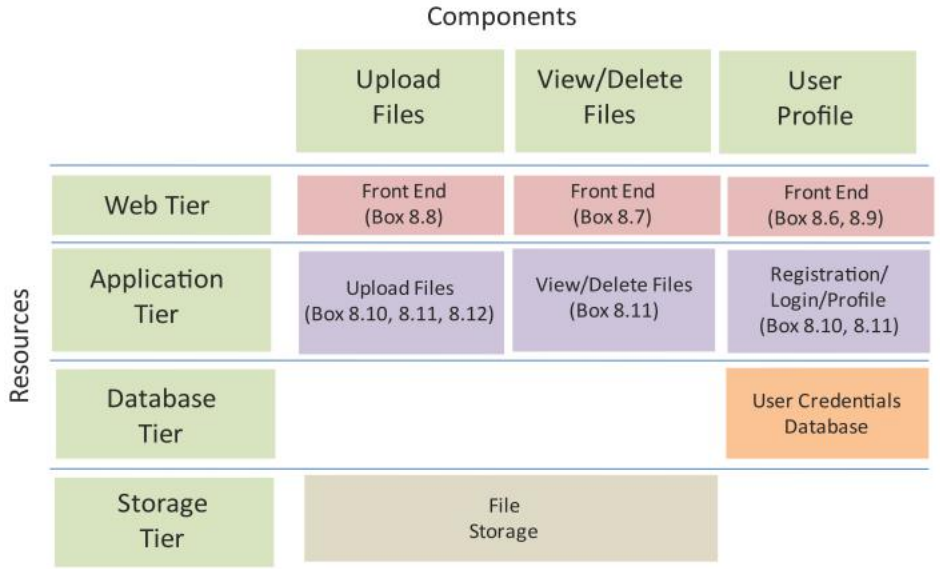
Deployment design for Image Processing App

Image Processing App Demo

Cloud Drive App – Component Design

- Functionality:
  - A cloud-based document storage (Cloud Drive) application.
  - This application allows users to store documents on a cloud-based storage.

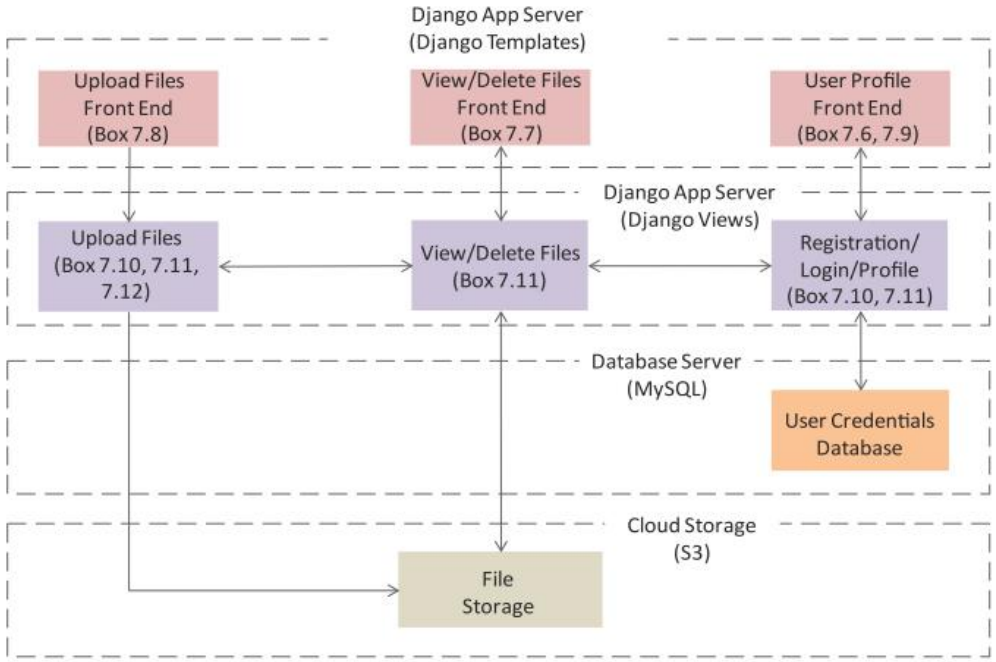
- Component Design
  - **Web Tier:** The web tier for the Cloud Drive app has front ends for uploading files, viewing/deleting files and user profi le.
  - **Application Tier:** The application tier has components for processing requests for uploading files, processing requests for viewing/deleting files and the component that handles the registration, profile and login functions.
  - **Database Tier:** The database tier comprises of a user credentials database.
  - **Storage Tier:** The storage tier comprises of the storage for fi les.



Component design for Cloud Drive App

Cloud Drive App – Architecture Design

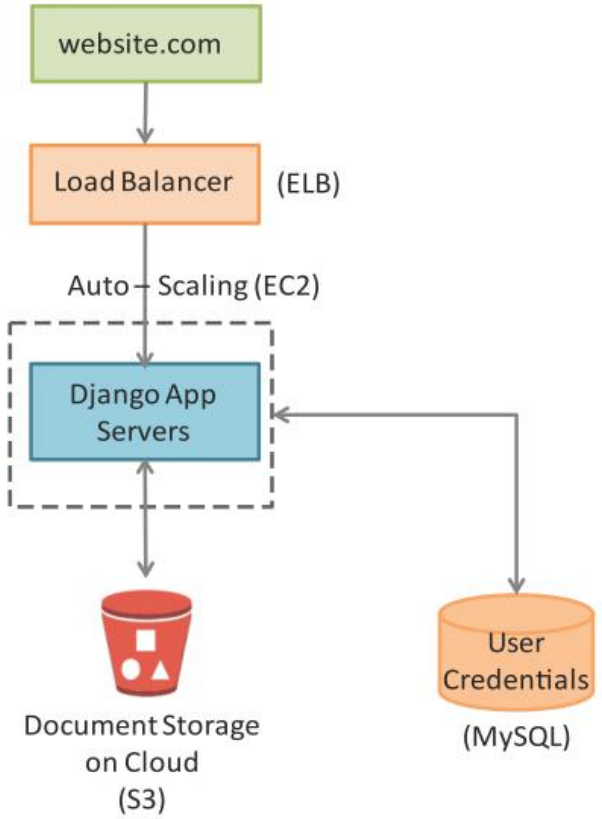
- Architecture design step which defines the interactions between the application components.
- This application uses the Django framework, therefore, the web tier components map to the Django templates and the application tier components map to the Django views.
- A MySQL database is used for the database tier and a cloud storage is used for the storage tier.
- For each component, the corresponding code box numbers are mentioned.



Architecture design for Cloud Drive App

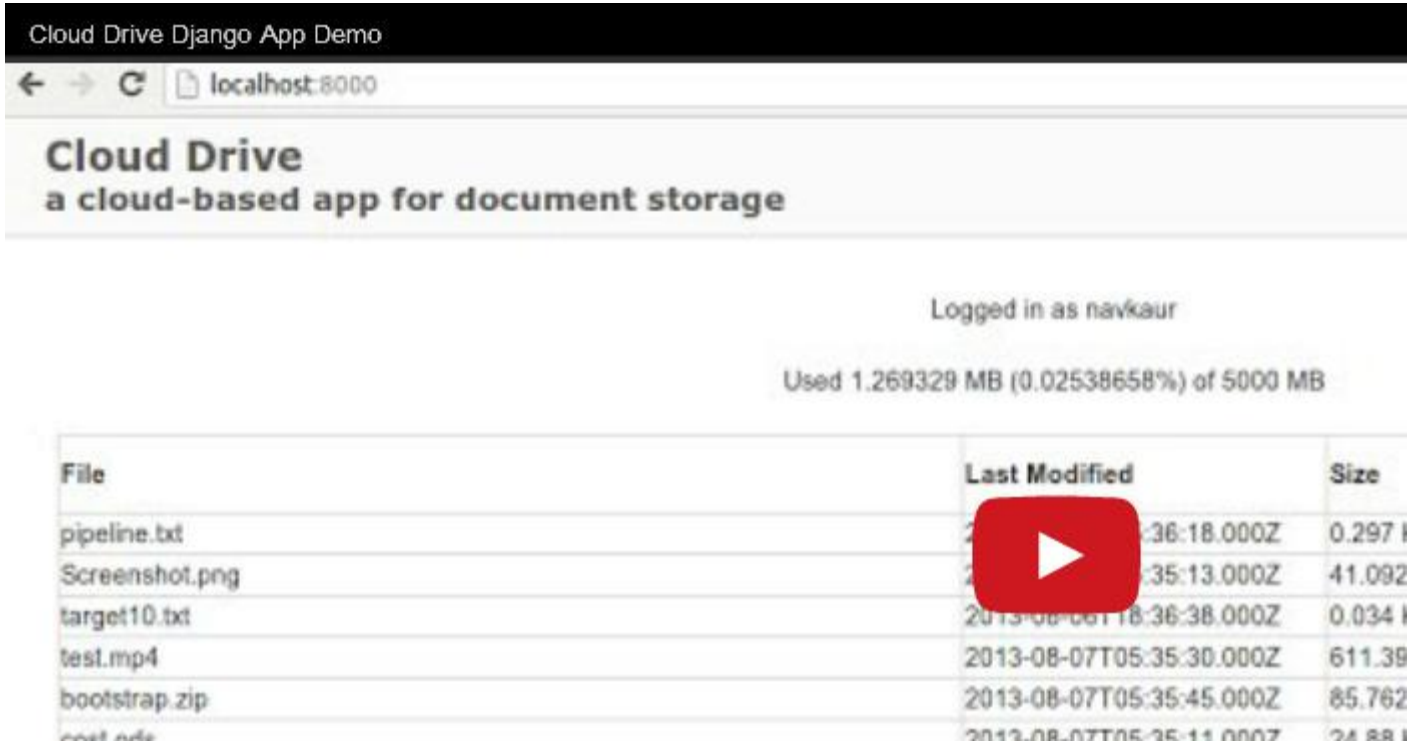
Cloud Drive App – Deployment Design

- Deployment for the app is a multi-tier architecture comprising of load balancer, application servers, cloud storage for storing documents and a database server for storing user credentials.
- For each resource in the reference architecture the corresponding Amazon Web Services (AWS) cloud service is mentioned.



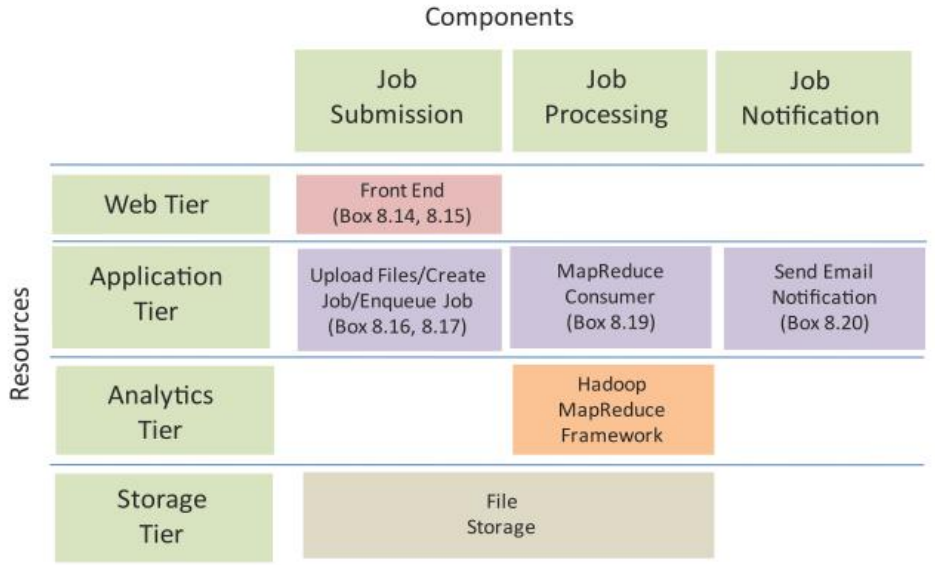
Deployment design for Cloud Drive App

Cloud Drive App Demo



MapReduce App – Component Design

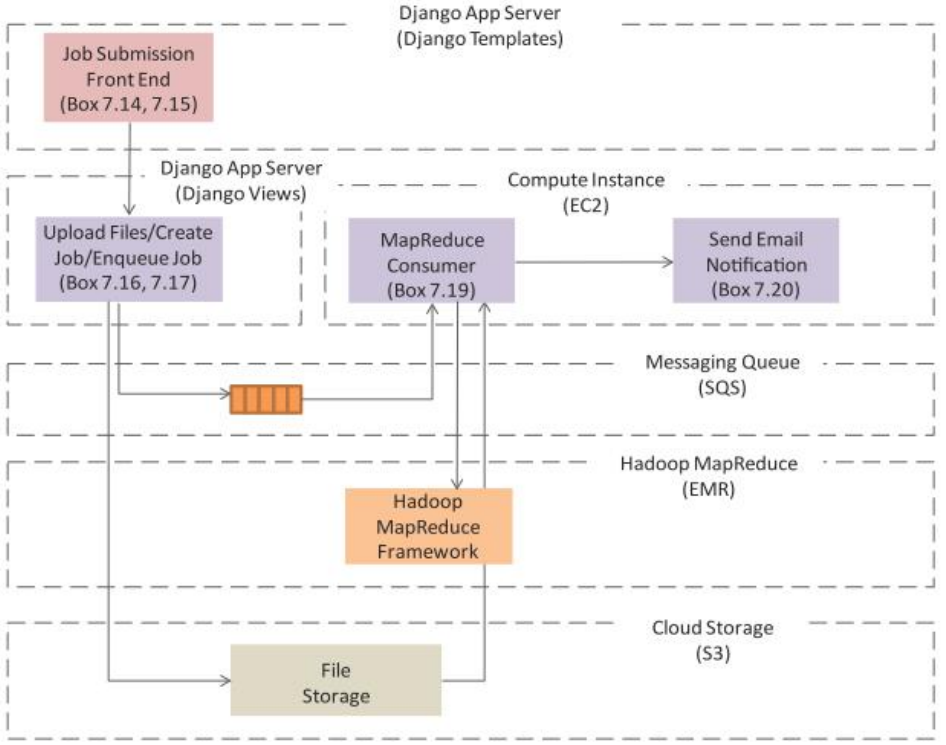
- Functionality:
  - This application allows users to submit MapReduce jobs for data analysis.
  - This application is based on the Amazon Elastic MapReduce (EMR) service.
  - Users can upload data files to analyze and choose/upload the Map and Reduce programs.
  - The selected Map and Reduce programs along with the input data are submitted to a queue for processing.
- Component Design
  - **Web Tier:** The web tier for the MapReduce app has a front end for MapReduce job submission.
  - **Application Tier:** The application tier has components for processing requests for uploading files, creating MapReduce jobs and enqueuing jobs, MapReduce consumer and the component that sends email noti fications.
  - **Analytics Tier:** The Hadoop framework is used for the analytics tier and a cloud storage is used for the storage tier.
  - **Storage Tier:** The storage tier comprises of the storage for files.



Component design for MapReduce App

MapReduce App – Architecture Design

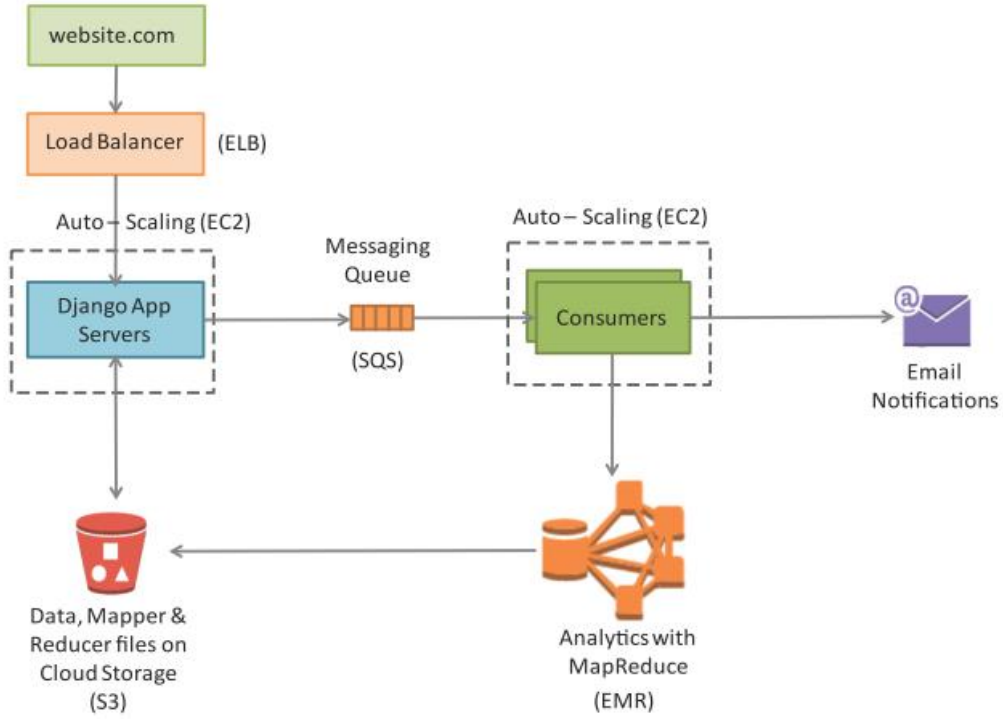
- Architecture design step which defines the interactions between the application components.
- This application uses the Django framework, therefore, the web tier components map to the Django templates and the application tier components map to the Django views.
- For each component, the corresponding code box numbers are mentioned.
- To make the application scalable the job submission and job processing components are separated.
- The MapReduce job requests are submitted to a queue.
- A consumer component that runs on a separate instance retrieves the MapReduce job requests from the queue and creates the MapReduce jobs and submits them to the Amazon EMR service.
- The user receives an email notification with the download link for the results when the job is complete.



Architecture design for MapReduce App

MapReduce App – Deployment Design

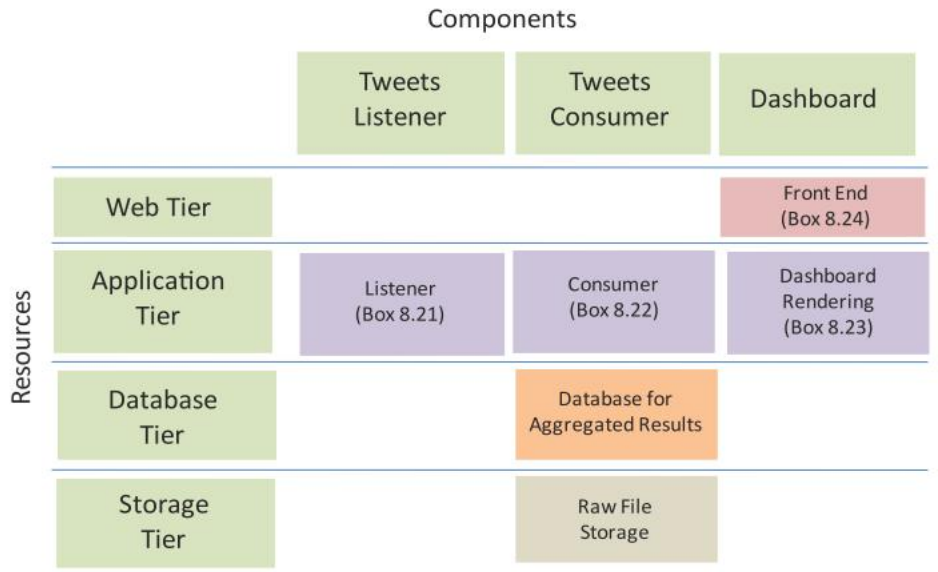
- Deployment for the app is a multi-tier architecture comprising of load balancer, application servers and a cloud storage for storing MapReduce programs, input data and MapReduce output.
- For each resource in the deployment the corresponding Amazon Web Services (AWS) cloud service is mentioned.



Deployment design for MapReduce App

Social Media Analytics App – Component Design

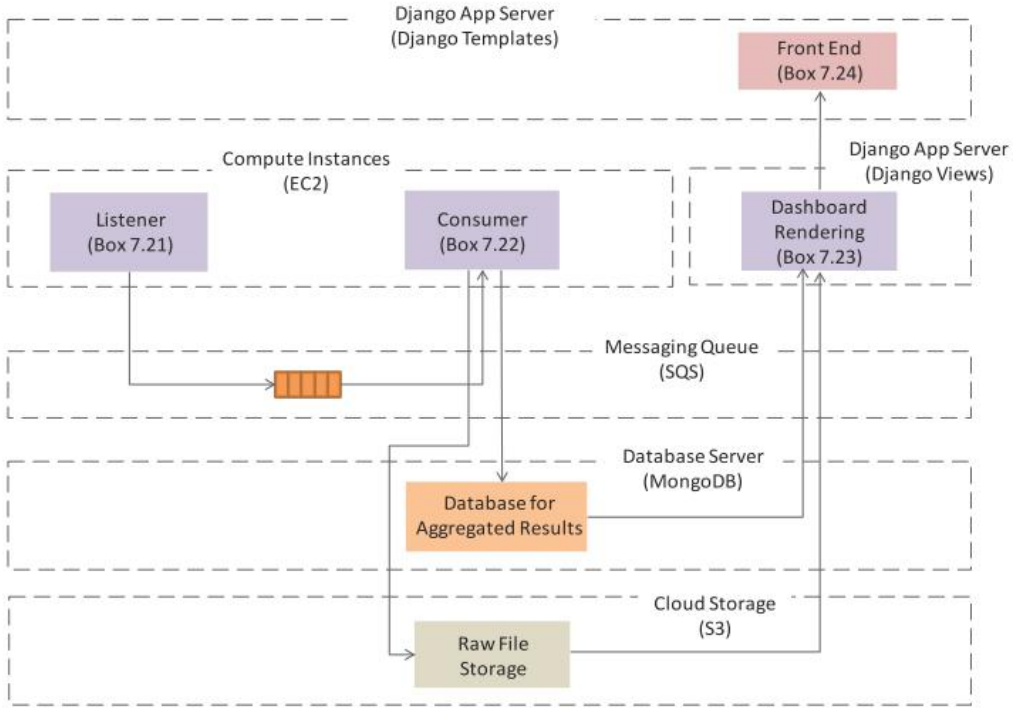
- Functionality:
  - A cloud-based Social Media Analytics application.
  - This application collects the social media feeds (Twitter tweets) on a specified keyword in real time and analyzes the sentiments of the tweets and provides aggregate results.
- Component Design
  - **Web Tier:** The web tier has a front end for displaying results.
  - **Application Tier:** The application tier has a listener component that collects social media feeds, a consumer component that analyzes tweets and a component for rendering the results in the dashboard.
  - **Database Tier:** A MongoDB database is used for the database tier and a cloud storage is used for the storage tier.
  - **Storage Tier:** The storage tier comprises of the storage for files.



Component design for Social Media Analytics App

Social Media Analytics App – Architecture Design

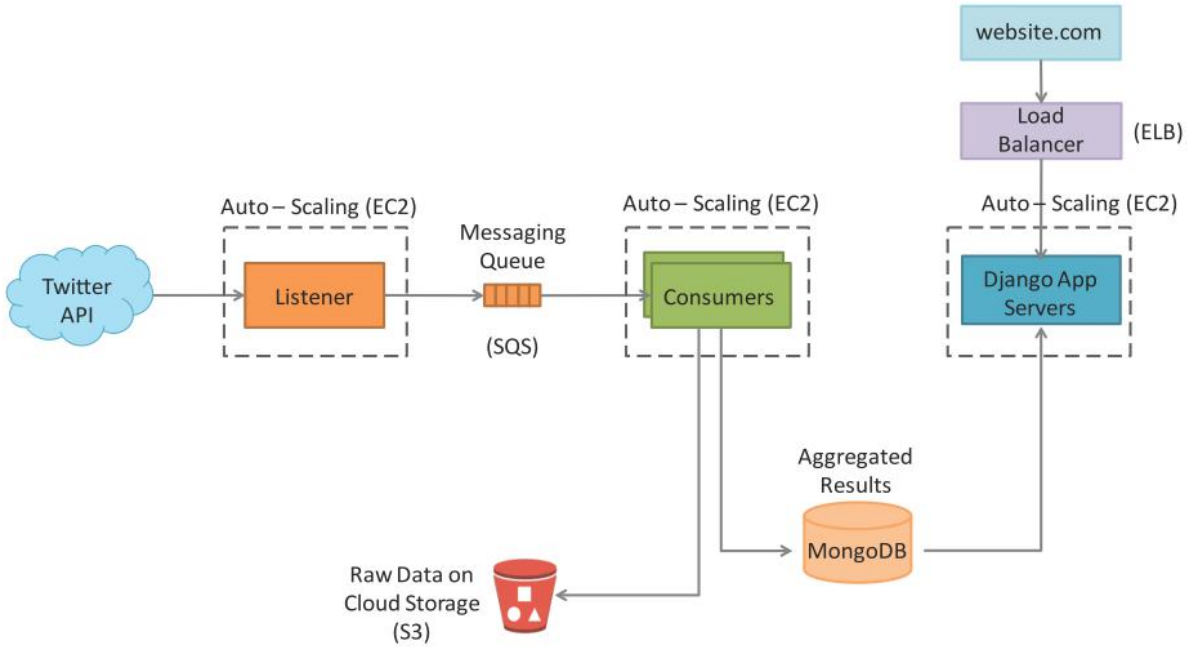
- Architecture design step which defines the interactions between the application components.
- To make the application scalable the feeds collection component (Listener) and feeds processing component (Consumer) are separated.
- The Listener component uses the Twitter API to get feeds on a specific keyword (or a list of keywords) and enqueues the feeds to a queue.
- The Consumer component (that runs on a separate instance) retrieves the feeds from the queue and analyzes the feeds and stores the aggregated results in a separate database.
- The aggregate results are displayed to the users from a Django application.



Architecture design for Social Media Analytics App

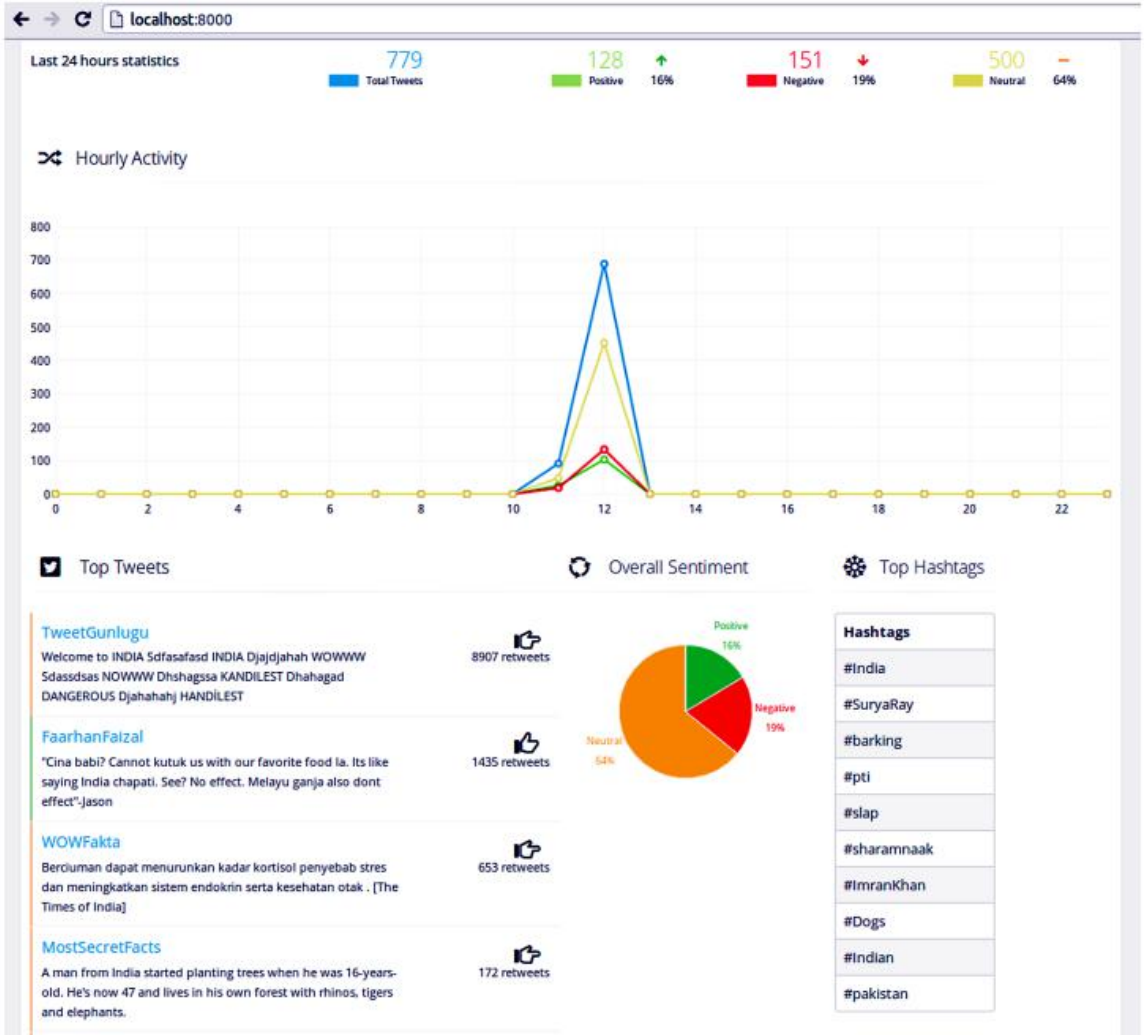
Social Media Analytics App – Deployment Design

- Deployment for the app is a multi-tier architecture comprising of load balancer, application servers, listener and consumer instances, a cloud storage for storing raw data and a database server for storing aggregated results.
- For each resource in the deployment the corresponding Amazon Web Services (AWS) cloud service is mentioned.



Deployment design for Social Media Analytics App

Social Media Analytics App – Dashboard



Further Reading

- A. Bahga, V. Madisetti, Rapid Prototyping of Advanced Cloud-Based Systems, IEEE Computer, vol. 46, iss. 11, Nov 2013.
- Amazon Web Services, http://aws.amazon.com, Retrieved 2013.
- boto, http://boto.readthedocs.org/en/latest/, Retrieved 2013.
- Django, https://docs.djangoproject.com/en/1.5/, Retrieved 2013.
- Django Models, https://docs.djangoproject.com/en/1.5/topics/db/models/
- Django Views, https://docs.djangoproject.com/en/1.5/topics/http/views/
- Django Templates, https://docs.djangoproject.com/en/1.5/ref/templates/builtins/
- Django URL dispatcher, https://docs.djangoproject.com/en/1.5/topics/http/urls/