Introduction

This document specifies the requirements the target environment used to host our application code to a remote cloud services platform. We define what artefacts are needed in the application in order to create an instance of the application

We detail the server configuration used to run the client side web application, database and load balancing technology used in our deployment.

Scope

The purpose of this stage of our development is to show our web application can be successfully run in a production environment on the publicly hosted site on the web. We discuss the underlying server configuration and applications that will be used to processes requests made by a user.

this ensures that the same process can be performed on each environment

Application Deployments define the package of software components that make up an application in a particular environment, e.g. development or production. Instances of these are deployed onto physical Technology Nodes to capture where that software is executing.

<https://www.w3.org/TR/2006/WD-WAPF-REQ-20061109/#ref-atomautodiscovery>

<https://vsupalov.com/how-backend-and-frontend-communicate/>

HTTP request/response payloads

requests are handled by a Nginex Reverse Proxy as the intermediary service used on the side of the server. This providers load balancing, TLS/SSL capabilities and acceleration

web server hands the request over to the app server where it is processed and sent a response

Flask is used A shared, operational environment as a Database server for hosting internal applications

A test environment

Backend Resonds with

HTML-formatted responses

other static files (CSS, JS, images, …)

JSON-formatted data

No body at all. Just a status code and header fields.

The frontend sends: Simple HTTP requests without a body Form data JSON-formatted data

Request for the backend arrive at the server and are eventually passed on to your backend code

Backend receives requests and prepares data which is transmitted back to the user’s browser.

Request handled by web server NGINX at the backend

GET request for a resource handled by backend response contains the file

web server hands the request over to the app server where it is processed and sent a response

server side technologies to render our website

generate an ennd point to access

(server configuration to run your web application, database and any load balancing services.)

To push our local development environment

So it is able to be accessed through a remote cloud services platform, to host our application code and

Deliver static and dynamic files quickly around the world using a Content Delivery Network to any client who requests them from anywhere on the web

Version that is publicly available to anyone

virtual machines in the cloud on which you have the OS level control.

scalable container service to allows you to run Docker containers in the cloud.

Still have the version on our local host that we can continue tweaking and testing while periodically updating the public code after testing

Responsive website

create an application package Perform a deployment to a target environment

define what artifacts are needed in the application and location of those artifacts

then creating an instance by pulling the needed content and creating an image of the application and storing it in repository

once an instance is created unique actions are performed dependent on the type of content to deploy the instance to each environment

this ensures that the same process is performed on each environment

different applications and there versions being deployed on to different environment