Module 12
DevOps

CS 169: Software Engineering

1 Overview

At this point, you and your team of software engineers are more than capable of deploying a full fledged Ruby on Rails web application with a JavaScript-enhanced, HTML/CSS based front end along with a Rails back end equipped with ActiveRecord. On top of that, you've continually added to your fantastic suite of tests to verify correctness and behavior. Give yourselves a pat on the back!

That being said, as the old adage goes, even "The best-laid plans of mice and men often go awry". No matter how carefully a project is planned, something may still go wrong with it. DevOps encapsulates a set of practices which compliments Agile, and is meant to help streamline software quality improvement. In this worksheet, we'll investigate different techniques for identifying and resolving post-deployment issues.

2 The Basics

Performance and security are the general focus of this module. Specifically, performance stability is qualified as responsiveness, resource management, scalability, and availability. On the other hand, security is a matter of privacy and authentication. Which aspects of application scalability are not automatically handled for you in a PaaS environment?

If your app "outgrows" the capacity of the largest database offered by the PaaS provider, you will need to manually build a solution to split it into multiple distinct databases. This task is highly app-specific so PaaS providers cannot provide a generic mechanism to do it

3 Three Tier Architecture

With the new revenue you've gotten from the application your team developed, you decide to purchase more devices to scale your application, developed in a shared nothing, three tier architecture format.

Out of the three layers, the Presentation and Logic tiers are much easier to scale than the Persistence tier. In max 2 sentences, explain how the shared nothing design helps with scalability, and what communication protocol + protocol's feature make this possible.

The Shared Nothing architecture is so called because "entities within a tier generally do not communicate with each other. Because of this, we can add computers to each tier independently to match demand" (ESaaS 12.2). In other words, adding more machines does not affect the performance of existing machines. HTTP's statelessness ensures requests can be treated independently and assigned to any machine, as opposed to a designated one.

4 SLO Calculations

During your last retrospective meeting, you and your team decided on an SLO regarding request latency: 90% of requests within any 1 minute window should have a latency below 1000ms. A couple days later, your profiling reports come back in 20 minute intervals, and you notice an average of 10000 requests are made during each 20 min interval.

What is the maximum number of slow or dropped requests (> 1000ms) that can occur before you fail to meet the SLO's standard? (Note: You do *not* know the timestamp of each request. However, you do know that at least 5 requests are being made each minute). In your answer, box your solution and show work.

Two cases to consider:

* Uniformly Distributed Requests across Time: 10000 requests / 20 min = 500 requests per minute. 500 requests per min * 0.1 = 50 slow/dropped requests per min at most. 50 slow/dropped requests per min * 20 min = 1000 requests total.

* Majority of requests occur during a single 5 minute time frame: Since min 5 requests occur per minute, all requests must not be slow/dropped (4/5 < 90%).

That means up to 10000 - 95 = 9905 requests could've occurred in a single 5 minute frame. Out of these 9, 9810 * 0.1 = 981 requests could have been slow/dropped.

* Therefore, 1000 slow/dropped requests is the maximum.

5 Feature Flag Configurations

Which of the following are appropriate places to store the value of a simple Boolean feature flag and why:

(a) a YAML file in the app's config directory
(b) a column in an existing database table
(c) a separate database table

Keep in mind, an app's source code shouldn't have to be modified for a feature flag.

The point of a feature flag is to allow its value to be changed at runtime without modifying the app. Therefore (a) is a poor choice because a YAML file cannot be changed without touching the production servers while the app is running.

6 Feature Flag Configurations

Which of the following key performance indicators (KPIs) would be relevant for Application Performance Monitoring:

(a) CPU utilization of a particular computer
(b) Completion time of slow database queries
(c) View rendering time of 5 slowest views.

Query completion times and view rendering times are relevant because they have a direct impact on responsiveness, which is generally a Key Performance Indicator tied to business value delivered to the customer. CPU utilization, while useful to know, does not directly tell us about the customer experience.

7 Databases

7.1

The textbook mentions that passing `:layout=>false` to `caches_action` provides most of the benefit of action caching even when the page layout contains dynamic elements such as the logged-in user's name. Why doesn't the `caches_page` method also allow this option?

Since page caching is handled by the presentation tier, not the logic tier, a hit in the page cache means that Rails is bypassed entirely. The presentation tier has a copy of the whole page, but only the logic tier knows what part of the page came from the layout and what part came from rendering the action.

7.2

An index on a database table usually speeds up ______ at the expense of ______ and ______.

Query performance at the expense of space and table-update performance

8 SQL Injection

Let's assume your application implements a login script using the following JavaScript code + SQL syntax:

```
var username = req.body.username;
var password = req.body.password;
var query = "SELECT␣*␣FROM␣Users␣WHERE␣username␣=␣'" + username + "'␣and
␣␣␣␣password␣=␣'" + password + "'";

db.get(query, function(err, row) {
    if (err) {
        // Login Fails
    } else {
        // Login succeeds as the user
    }
}
```

The above code is vulnerable to SQL Injection!

8.1 Let the Table Drop

Describe what input could be passed in for `username` and `password` to exploit this vulnerability such that you would drop the `Users` table. Assume that "Mallory" is a valid username.

1. Set the value of username: Mallory
2. Set the value of password: 0'; DROP TABLE Users; –

8.2 Identity Theft is not a Joke, Jim!

A jealous user knows that there is a rich user with `username` "Dwight".

Describe what input could be passed in for `username` and `password` to exploit this vulnerability such that one could log in as "Dwight" *without* knowing Dwight's PIN.

1. Set the value of username Dwight
2. Set the value of password 0' OR '1'='1