pdx.edu migration

2023-03-02

Architecture Overview

CloudFront distribution

https://aws.amazon.com/blogs/networking-and-content-delivery/secure-and-accelerate-drupal-cms-with-amazon-cloudfront-aws-waf-and-edge-functions/

https://github.com/aws-samples/amazon-cloudfront-secure-accelerate-drupal

Viewer/admin distribution

- caching distinctions
- https://drupal.web.wdt.pdx.edu v. https://drupal-admin.web.wdt.pdx.edu

S3 backed

- *public* bucket
- -

DNS/zone and cert handling

- e.g., update stack with www.pdx.edu zone

  request (email validation) cert ahead of time to use during development and through changeover. upon changeover, add DNS-validated cert, update site during regular maintenance

Worker architecture, configuration

https://github.com/awsdocs/elastic-beanstalk-samples

https://github.com/aws-samples/eb-php-drupal

- “bursty” architecture
- HA architecture

Filesystem backend

- HA (internal)

- supports worker HA

Database backend

- “bursty” architecture
- HA architecture

TODO

we have fairly rigorous caching configured on the frontend, but what about caching on the backend?

- varnish instance?
- elasticache instance?
- opcache?

Benchmarking / Load Characterization

In order to right size for both performance and development/maintenance workloads, we need to characterize these loads.

pdx.edu (performance)

- "frontend" performance
    1. test using 'bursty' architecture; this approach has a significantly cheaper baseline cost, but is a modality that won't suit more "constant load" workloads.
    2. is pdx.edu a bursty or constant load?
    3. do we need to increase / can we get away with the burst quotas to serve the, e.g., 90/95th percentile requests?
- backend performance
    1. again, test using 'bursty' architecture
    2. db currently configured HA with constant IOPS
    3. blue/green deployments? https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/blue-green-deployments-creating.html
    4. can use RDS performance metrics!

pdx.edu site import, migrations and deployments (dev/maint)

- test using 'drupal-manager'
- does the box itself have the resources required to run deployments/migrations (drush)
- does this approach have any deficiencies; do we have hard constraints or preferences to use, e.g., a queueing mechanism?

Policy

Not urgent, but we should develop/record a robust/rigorous policy that captures expectations surrounding uptime, availability and maintenance.

Worker

- maintenance window

Database

- maintenance window
- backup window

References

https://github.com/aws-samples/aws-refarch-drupal

https://portlandstate.atlassian.net/wiki/spaces/WEBCOMM/pages/2522152988/Drupal+in+AWS+Elastic+Beanstalk+Proof+of+Concept

https://aws.amazon.com/blogs/security/hardening-the-security-of-your-aws-elastic-beanstalk-application-the-well-architected-way/