I was unable to fully complete the exercise in the time I had available, however following is my solution:

The architecture I chose was to use an Eventbridge rule to trigger a Lambda script to create the file in s3 every ten minutes. The s3 bucket was configured to use a KMS Customer Managed Key to encrypt all contents. The web interface was provided by producing a Cloudfront distribution which, had I completed the task, would have needed a Web Application Firewall.

The rationales for these choices were:

- Eventbridge is the service which can cause events to happen on a schedule.
- Lambda because producing a timestamp file is a very simple thing to do, does not require compute infrastructure, and is free to use up to a certain threshold.
- s3 because it provides very inexpensive storage and can be used to make content available on the web.
- Cloudfront because when s3 contents are encrypted with KMS CMK's, it is required to decrypt the file contents before serving. This is done using a Lambda@Edge script attached to the Cloudfront distribution. Additionally, it is considered best practice to not expose s3 buckets or EC2 instances directly to the public internet, but to have their serving capability mediated through either Cloudfront or Load Balancers.
- WAF because it provides the capability to limit incoming traffic. WAF is also used for a great many other things, of particular potential interest here would be the ability to limit access to certain specified IP addresses, though this can also be done with s3 bucket policies.

This is a very inexpensive solution. According to the AWS cost estimator, this configuration should cost slightly more than $12 per month, almost all of which is WAF usage. More will be said in detail on this further down.

Assuming the users of this solution aren't using a third party monitoring solution such as Datadog, monitoring of all elements would be done primarily through AWS Cloudwatch metrics and logs, with the potential to configure events to send alerts through a variety of means (email, SMS notifications, etc.) if operation goes out of desired bands. Additionally, WAF activity can be logged in detail, with the data sent to S3.

Recovery from a regional disaster is generally done by deploying a parallel infrastructure in a different AWS region. If this is done as a "hot standby" with the same infrastructure deployed in multiple AWS regions, with the Cloudfront distribution accessing the services through a load balancer, the impact to end users in nonexistent in the event of a single region issue, so long as the problem is not a systemic one with either WAF or Cloudfront. If this service is not considered very critical, it could be quickly deployed via Terraform within a few minutes.

As far as compliance with the AWS Well Architected Framework. Several elements of this are represented in this design, the first being deployment solely with Terraform, thus performing operations as code. I am of the belief that no deployment operations in environments higher than development should be done using the console, that Infrastructure as Code should be strictly enforced. This design is not capacity constrained, as no servers are involved, and all elements of it are managed by AWS to perform at very high levels. Also, the data is encrypted both at rest and in transit, providing the maximum data security. Also, IAM roles which govern the access between AWS services, should be configured to give "least required access". This is not perfectly implemented here.

This repo is incomplete (the WAF is not included) due to my running out of time, and having extreme difficulties getting the Cloudfront distribution to decrypt the file in the s3 bucket. Getting this to happen is shockingly difficult to find useful information on, with what I was able to find being both old and of poor quality. Doing a web search for several hours trying to solve this problem, what I ran into frequently was, when someone asked how to do this in a forum, most of the answers they got were effectively, "why are you bothering to encrypt an s3 bucket you're serving as a web page?" I did find a repo in GitHub which was supposed to have a canned solution to this here: https://github.com/kurianoff/cloudfront-s3-lambda-at-edge but when deployed, it doesn't work either. What I do notice from this and other solutions I found is that they all implement Lambda@Edge with the Cloudfront distribution to decrypt the file, but the associated IAM role does not have a "kms:decrypt" rule. I did try adding "kms:*" to the rules at one point, but that also failed to get the distribution to decrypt the file. Something else is clearly missing which I couldn't find any documentation on despite hours of trying. I simply ran out of time to get this properly implemented.

Finally, the issue of cost. Again, this is using the AWS Cost Estimator. The cost of creating and storing the file comes in pretty much any case to $0.04 per month. The assumptions here are 5000 file creations per month (rounded up), Lambda being allocated 128 MB of memory and the average run taking 350ms. S3 is set to distribute 1 GB per month, which is ample, as the file created is roughly 20 bytes. This does assume Lambda is still in the free usage tier. These should be insensitive to usage. The main contributors to cost differential are Cloudfront and WAF. For Cloudfront, in summary, if the file is downloaded 1000 times per month, the cost is $0.09; if 10,000 times per month, it goes up to $0.10; if 100,000 times, $0.19, and one million, $1.09. Bear in mind that for all of these cases, total data transfer out is still the minimum of 1GB per month chargeable, however the first 1024 GB per month is free. For the WAF, this is more complicated. I assume the WAF for this only requires one web ACL, and I assume 10 rules, which seems a very conservative estimate. That comes to $15 per month, which completely swamps out all other considerations unless the site produces massively more than one million impressions per month.