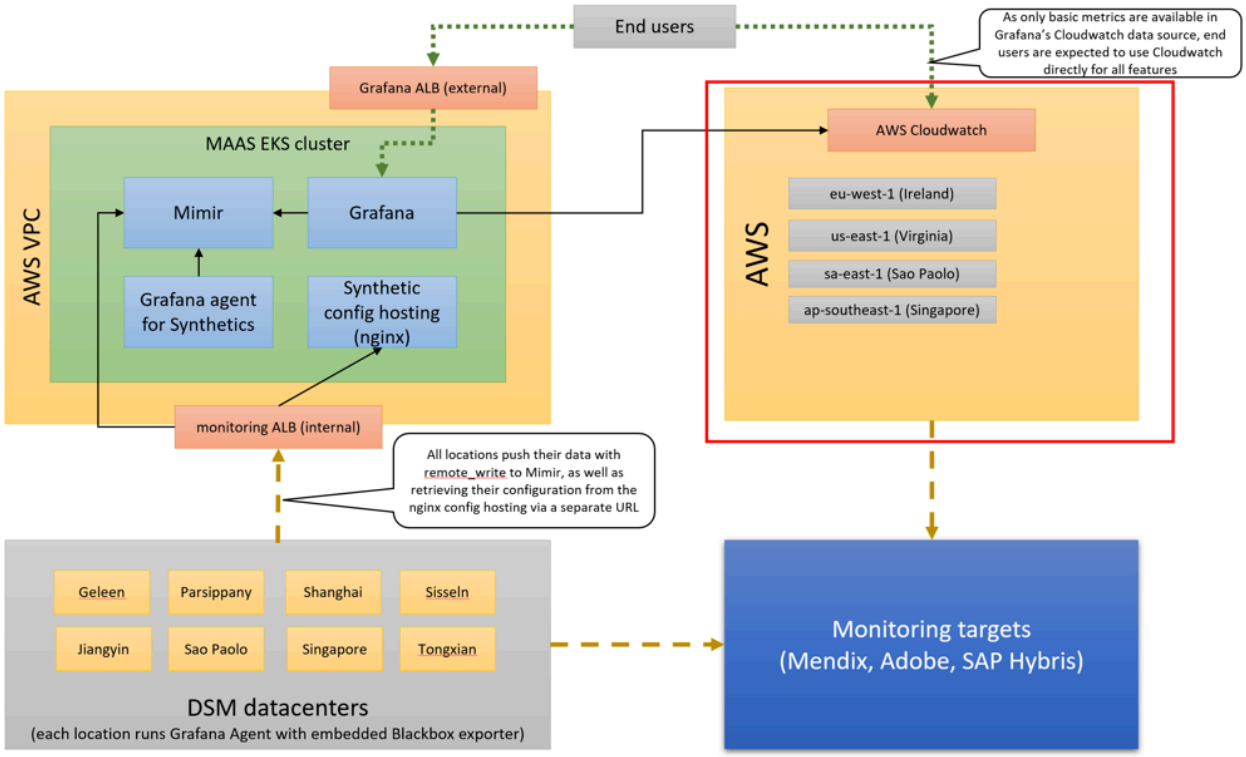
CloudWatch Synthetics (Browser / Clickpath based)

The synthetic monitors are continuously monitoring a given set of websites in a set interval and report metrics such as page load times, time to first byte, transfer times, etc. Within the synthetics space we define two types of monitors; HTTP based and browser based. HTTP based is a simple HTTP call, usually an HTTP GET, and checks the HTTP status code along with response times. Browser based, or clickpath based as it was known in Dynatrace, utilize a headless browser to perform the same actions an actual user will be doing. It uses a Chromium-based browser and, unlike a HTTP-based monitor, also renders JavaScript and allows you to programmatically interact with the website itself. Browser based monitors are more advanced and offer a deeper level of monitoring than HTTP based monitors will be able to do. As you read in the parent page, we use CloudWatch Synthetics for browser-based monitors and Grafana Agent with the blackbox exporter plugin for the HTTP based ones. The latter also runs on DSM private locations at this time of writing, the former only runs in AWS regions. In the future we will look if we can add Grafana K6 as part of browser-based stack.

> This page focuses solely on the CloudWatch Synthetics portion of the stack; please review the other pages for Grafana Agent based monitoring.



CloudWatch Synthetics runtime & development notes

CloudWatch Synthetics utilizes Puppeteer to perform its browser-based checks. They are pure JavaScript based and the monitors themselves execute as a NodeJS Lambda which runs in an AWS region. At this time of writing the DSM AWS Team enabled 4 regions on our AWS accounts so we check a single website from up to 4 locations (this is configurable per website, see below for more details). As it uses a Lambda, the logs of the execution are inherently captured in CloudWatch Logs, and other resources such as screenshots and HAR files are stored in S3 for every execution of the Lambda. Each region has a single dedicated S3 bucket to store all of these artifacts for all monitors that run in this region. The screenshot functionality of CloudWatch Synthetics also supports variance detection, but I was unable to get this to work properly (it always detected more than 50% of changes to the page even though the page was visually the same). This is disabled and can be re-enabled in a future iteration.

CloudWatch Synthetics also has a browser plugin that allows you to record interactions with the browser and generate JavaScript out of this. This is generally useful as a starting point but typically not sufficient for an end-to-end test. The JavaScript is not always valid and using it as-is can lead to flaky results or simply not even working. You will likely have to modify the generated JavaScript code to make it work. When I was creating these monitors I usually used the browser recorder to get a basic boilerplate, then used vanilla Puppeteer to quickly iterate over the JavaScript code and, once it worked in Puppeteer, embed that JavaScript code into the AWS Synthetics wrapper code and test it from the Lambda. Most of the time this produced a valid response, but I had one case where the code worked in vanilla Puppeteer but not in CloudWatch Synthetics. There are also known cases that a monitor doesn’t work in headless mode, but this is not something I experienced myself. It takes effort to develop these monitors with a lot of trial-and-error, so ensure you allocate sufficient time for creating custom monitors.

Enabled regions

The AWS account is controlled by the DSM AWS team and has 4 regions enabled. Those are:

- eu-west-1 (Ireland)
- us-east-1 (Virginia)
- sa-east-1 (Sao Paolo)

- ap-southeast-1 (Singapore)

So these are also the regions the AWS Synthetics use by default (which is configurable as you can see in the next section). There is technically no reason to not offer this for other regions, but it requires the AWS team to enable this region on our AWS accounts. There is currently an explicit deny on all other regions so they are inoperable.

Infrastructure-as-code

The creation of the underlying infrastructure is all automated with AWS CDK code and has a pipeline. The code lives in a separate repository called *aws-synthetics*. This codebase has a couple of concepts it introduces:

- configuration: YAML files for each CloudWatch Synthetic monitor, which is required for every monitor you define. The CDK code dynamically creates resources based on this configuration YAML file so you don't need to add additional CDK code to add new monitors.
  - If you need configuration at runtime or things like passwords you should embed that in the JavaScript code to query the SSM Parameter Store or Secrets Manager at runtime to retrieve it.
  - Environment variables can be added via the `envVars` option in the YAML file. This is optional.
- type: The type of monitor. Can be `custom` or `synthetic` At this time of writing all monitors are of type custom.
  - The synthetic type is a generic monitor that can be reused. The idea is that if there common patterns that you want to reuse (let's say for Mendix applications) you can utilize a synthetic monitor type with just different input parameters. At this type of writing this is not used but the underlying code supports it.
  - The custom one is a custom monitor with its own JavaScript source code which will be put verbatim in the executing Lambda.
- Execution time: Every monitor executes every hour. This is not configurable at this time of writing.
- Execution delay: Each region triggers the run of a monitor with a 3 minute delay. This to prevent all regions running the same check concurrently (this causes problems with CyberArk, for example).

- There is no concept of alarms in the codebase, which is done on purpose. Alarms are expected to be created in Grafana using the CloudWatch integration.

By default, a YAML configuration only has a couple of required parameters:

```yaml
# The type of monitor to use. Valid values are 'synthetic' or 'custom'. Use synthetic for commonly used patterns,
# custom to write your own javascript code or generate it using the AWS Cloudwatch Synthetic Recorder plugin for
# Chrome. Generated code by the recorder can be further modified afterwards.
type: custom

# Is this monitor enabled or disabled
enabled: true

# A list of AWS regions to execute this check on
regions:
  - eu-west-1
  - us-east-1
  - sa-east-1
  - ap-southeast-1

# The name of the source code file to use, found in the 'monitors' folder.
source: adobe_dyneema.js

# Optional environment variables
envVars:
```

All four parameters mentioned here are required, with the envVars being optional. You can add your own configuration but does require you to add support for it in the underlying CDK code.

Adding a new monitor

Just as a small recap, these are the steps you need to take in order to add a new monitor:

1. Determine the type of monitor you want to add. If you have a generic pattern that already exists as a synthetic monitor, consider using that. Otherwise, develop your custom one (we assume you want this in this example).
2. Develop your custom monitor JavaScript using the steps mentioned in the development notes section above. Validate it works with vanilla Puppeteer first and then embed this code as a valid AWS Synthetics JavaScript code.
3. Place your JavaScript code under `monitors/custom/` in the repository
4. Create a `configuration/<my-monitor-name>.yaml` with the layout mentioned above. The 4 required parameters must be in. Set the value of `source` to the value of your newly created JavaScript filename.

5. Run the pipeline to get your monitor created by the CDK codebase and await its execution.

Grafana integration

As CloudWatch Synthetics publishes metrics in CloudWatch itself, it becomes possible to view these metrics in Grafana as well. Note that metrics and raw Lambda execution logs are the only thing you can see in Grafana, and tihngs like screenshots and HAR files are not visible in the Grafana interface. This is also why it is recommended for users consuming this monitoring data to have access to the CloudWatch console. It is also expected that alarms are not created as CloudWatch alarms but instead in Grafana directly. Like this there aren't multiple mechanisms or sources generating alarms.

Further reading

- aws-cdk-lib.aws_synthetics module · AWS CDK (amazon.com)
- Using synthetic monitoring - Amazon CloudWatch