**Use case 2**

There is a pipeline which generates an app for windows. One of the steps in the

pipeline is to code sign the libraries and executables that are generated during the

build process and for each build it would generate around 100 files that would be

sent to the signing server for signing. This process works fine. But, when there are

multiple parallel builds say around 15 builds kicked-off simultaneously then the

signing process fails.

Given the high level requirement, come up with a proposal to redesign the system to

avoid failures.  
  
**Suggestions:**  
  
To redesign the system to handle multiple parallel builds and avoid failures during the signing process, here's my little suggestion for that :  
  
First thing we can do is made the scaling of the resource meaning that assiging the multiple server to handle the load.  
  
**Flow :  
  
Build Process File Paths -> Message Queue -> Signing api’s  
Load Balancer -> Multiple Signing Servers**  
  
**Multiple Signing Servers**:  
  
Set up multiple signing servers, each capable of handling a subset of the signing workload.  
These servers should be identical in terms of configuration and access to signing creds.  
Distribute incoming signing requests across these servers.  
Each server can process a subset of files concurrently.  
By using the auto scaling also we can do scale the resource

We can check whether the signing apis requests are working correctly on the servers

**Message Queue Integration**:  
Integrate a message queue system like RabbitMQ into the pipeline.  
After the build process completes, instead of directly sending files to the signing server, enqueue file paths in the message queue.  
  
**Load Balancer**:  
Implement a load balancer to distribute incoming signing requests among available signing servers.  
This ensures an even distribution of workload.   
  
**Redundancy**:  
Have redundancy in place. If one signing server fails, the load balancer should redirect requests to other available servers.  
  
**Monitoring and Alerts**:  
  
Implement monitoring for signing servers and the message queue.  
Alerts should be set up to notify respective team in case of any issues.  
For monitoring and alerting we can use  
**Monitoring**: Prometheus with Grafana for monitoring and visualization  
**Alerting**: Prometheus Alert manager or integrated tools

**Parallelism at the Build Stage:**

If possible, parallelize the build process itself.  
Generate fewer files per build by optimizing application dependencies  
Smaller builds reduce the load on the signing process.