

**CollaBoard**

Architecture at glance

Architecture at Glance

|  |  |
| --- | --- |
| Title: | CollaBoard Architecture at glance |
| Abstract: | This document details the specification for CollaBoard architecture on Azure and on-premise |
| Status\* | In Progress |
| Document Owner: | IBV |
| Document Author: | Gian Paolo Santopaolo |
| \*Draft, In progress, Reviewed, Authorized, Approved, Completed | |

|  |  |  |
| --- | --- | --- |
| **Document Control** | | |
| Document Distribution: | Internal: | Gian Paolo Santopaolo, Andreas Gorog, Michael Gorog, |
|  | External: |  |
| Document Authorization: | Authorizer (s): |  |



|  |  |  |  |
| --- | --- | --- | --- |
| **Change History** | | | |
| Date | Ver | Changed By | Details |
| 26/02/2018 | 1.0 | GP Santopaolo | Initial Draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Index

[Introduction 4](#_Toc507455480)

[Purpose of the document 4](#_Toc507455481)

[Foreword 4](#_Toc507455482)

[CollaBoard Azure architecture 5](#_Toc507455483)

[Standard Web API request 5](#_Toc507455484)

[Real-time data exchange 6](#_Toc507455485)

[File upload download 6](#_Toc507455486)

[Telemetry data 6](#_Toc507455487)

[CollaBoard on-premise architecture 7](#_Toc507455488)

[Custom IBV Managed File Transfer 7](#_Toc507455489)

[File Download: 8](#_Toc507455490)

[File Upload: 8](#_Toc507455491)

[Custom Telemetry 9](#_Toc507455492)

[Scaling considerations 10](#_Toc507455493)

# Introduction

## Purpose of the document

This document details the architecture used in CollaBoard running on Azure as well when it runs on-premise

One of the purposes of this document is to describe CollaBoard architecture in any possible variation customer can ask for CollaBoard running on IBV’s Azure, on customer’s Azure, running on customer data center like Azure Stack or fully premise on customer data center.

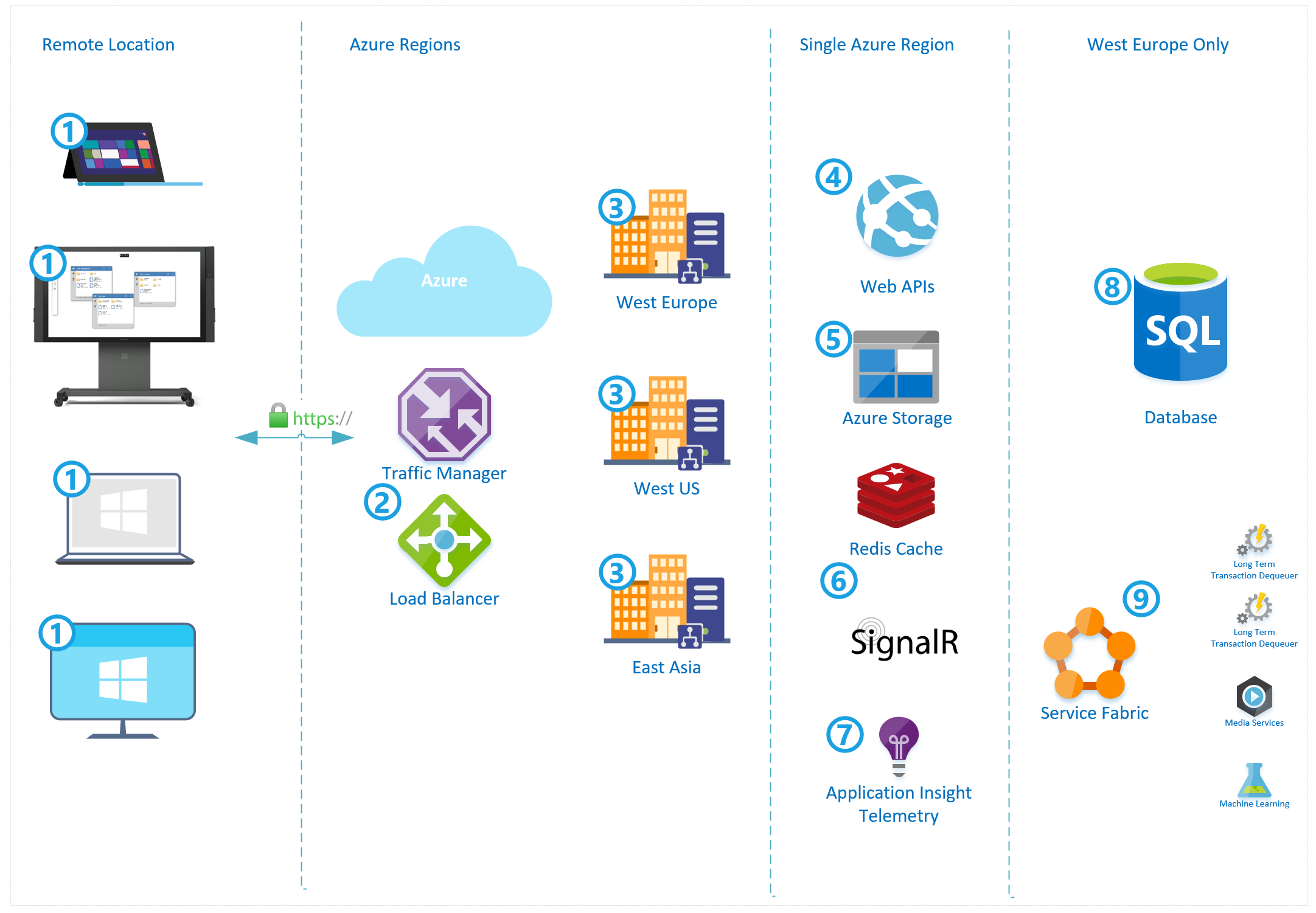
# Foreword

All the connection described in this document are secure connection (https)

# CollaBoard Azure architecture

In general, when a CollaBoard client (1, remote location) request a connection to Azure (Azure Regions) the traffic manager together with the load balancer (2) decides to Azure Region (3) send the request.

The request is sent to the nearest Azure Region. It can happen that because of high traffic or failure of a region a different one is chosen to guarantee client integrity.



*Fig.1 CollaBoard Architecture on Azure at glance*

When a CollaBoard client deals with services it can perform several service request, that can be categorized in:

1. Standard Web API request
2. Real-time data exchange
3. File upload and download
4. Telemetry data

## Standard Web API request

Client (1) performs a service request to the Web API (4) that is being processed by the business layer, data are collected or stored into the database (8), and a response is sent back to the client (1).

In some cases, standard Web API (4) request can originate data to be stored that will be analyzed by Service Fabric (9)

## Real-time data exchange

Client (1) is always connected with services through a Real-Time architecture (6) based on WebSockets. Thanks to this approach all the clients participating in the same projects can send and receive data to different clients.

The same data, when needed and when the business layer decides that is appropriate, are also stored in the database (8) or sent to Service fabric (9) for offline processing

## File upload download

Client (1) can upload and download files of any type and size. When uploaded files are sent from Client (1) to Azure Storage (5) through the Azure SDK that cares about all the process of chunking sending with retries, recreate the file on the server and store it on a Blob Storage.

When downloaded otherwise process is performed

## Telemetry data

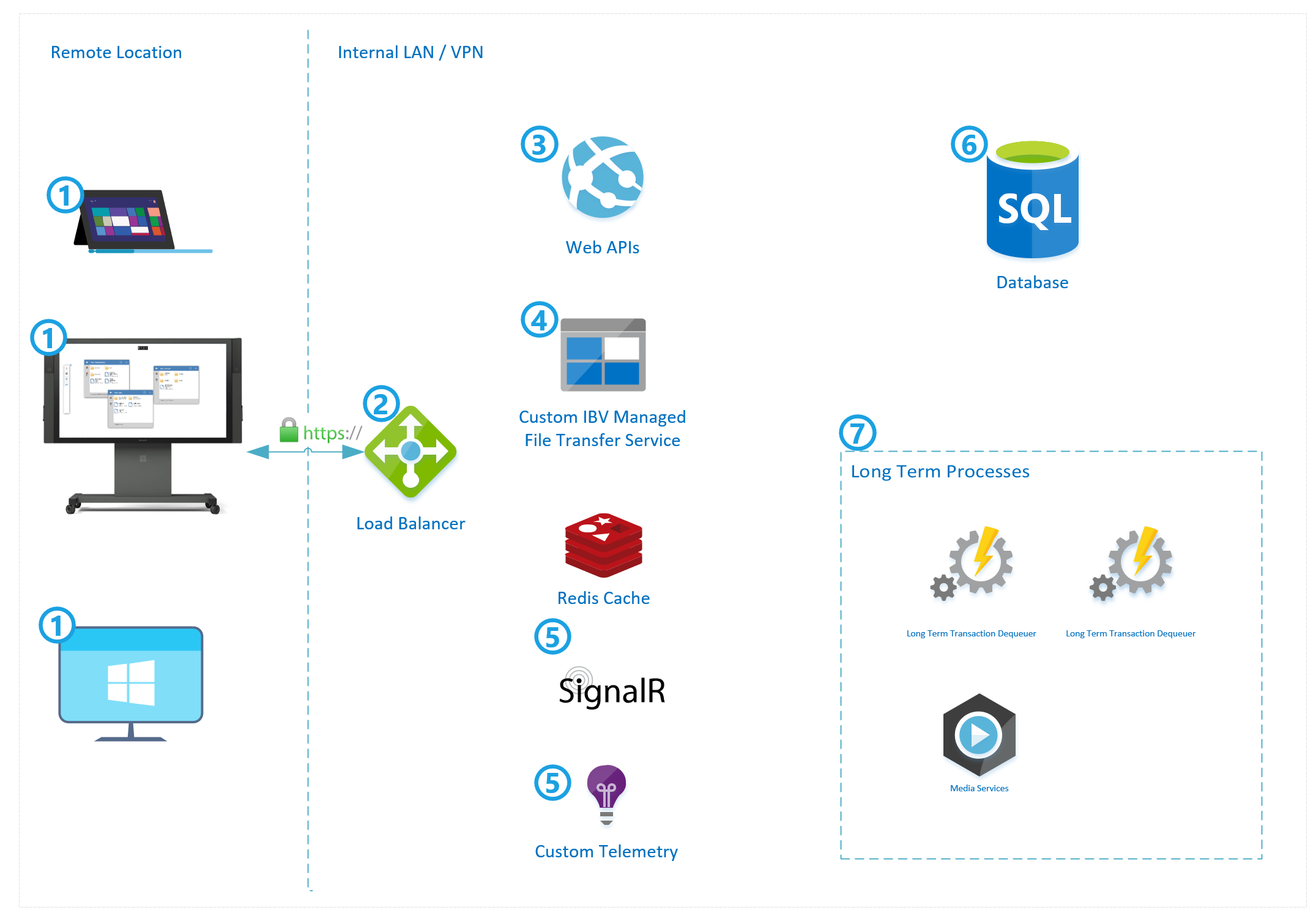
Each action performed on the Client(1) and server are stored on the Application Insight Telemetry (7) we have two different Telemetry category:

1. Application logs, including exceptions
2. Performed actions stored in an anonymous form (e.g., upload action, project creation action and so on)

Any Business Intelligence can later analyze these data.

# CollaBoard on-premise architecture

When CollaBoard is running on premise, is decoupled from Azure technologies, to achieve this goal we had to rewrite the Azure Storage component (5 in Fig. 1 CollaBoard Architecture on Azure at glance) and the Application Insight Telemetry (7 in Fig. 1 CollaBoard Architecture on Azure at glance)



*Fig.2 CollaBoard Architecture on-premise at glance*

Service Fabric (9 in Fig. 1 CollaBoard Architecture on Azure at glance) is running under the form of Long-Term Process (7)

Like on Azure as well as on-premise, a CollaBoard client (1, remote location) request a connection to the load balancer (2, internal LAN/VPN). It decides to send the request to the appropriate Web API (3), Custom Managed File Transfer (4), SignalR (5) or Custom Telemetry (5).

The request is sent to the nearest Azure Region. It can happen that because of high traffic or failure of a region a different one is chosen to guarantee client integrity

## Custom IBV Managed File Transfer

As already said one of the differences with the Azure architecture is file storage. On Azure, we were using the blob storage coming along with Azure storage SDKs for both client and server. This approach guarantees not to care about all the process of upload and to download file of each type and size.

To decouple totally from the Azure, we had to develop our own custom Managed File Transfer (4) that it consist of two main processes a file download and a file upload.

### File Download:

The user will ask for the list of his file; then it can choose which one to download.

A download request to the server is performed by the user (1). The request will also contain the chunk size client prefers.

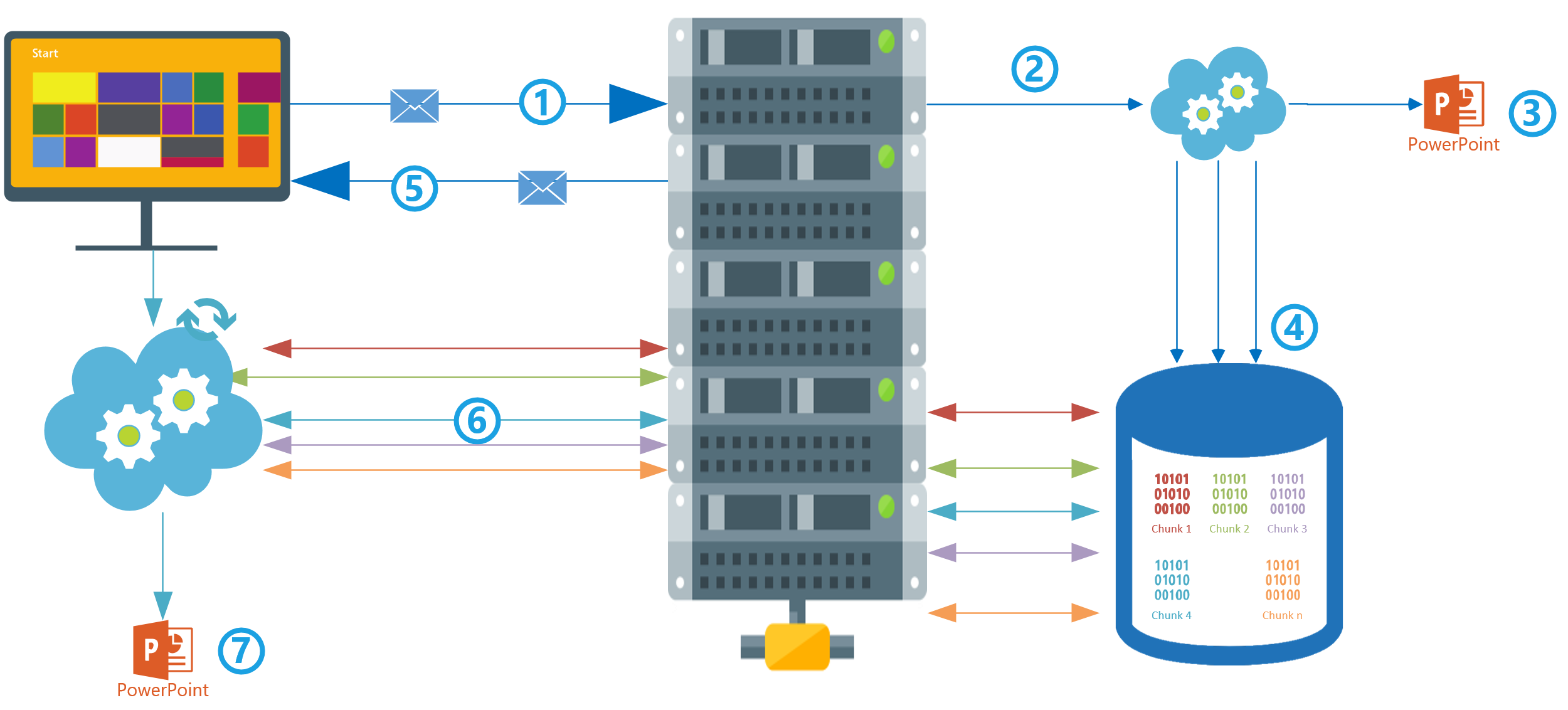
The server starts looking for the file (2) and, if found, it will be split in the appropriate number of chunks and saved into a common repository (4), so that solution is also load balancer compliant.

Once the chunking process finished, the client is notified so that it can start the download (5)

Client SDK starts downloading each chunk thanks to a state machine (6). The state machines internally will provide all the necessary architecture to download single chunks, retry to download if necessary and so on.

User can pause or cancel the process anytime (thanks to the cancellation token)

Once the client SDK is done, it will calculate the file checksum again, if this is the same sent from the server then the file is successfully downloaded (7) it will notify the client, thanks to an event, that file has been downloaded and available for being used.

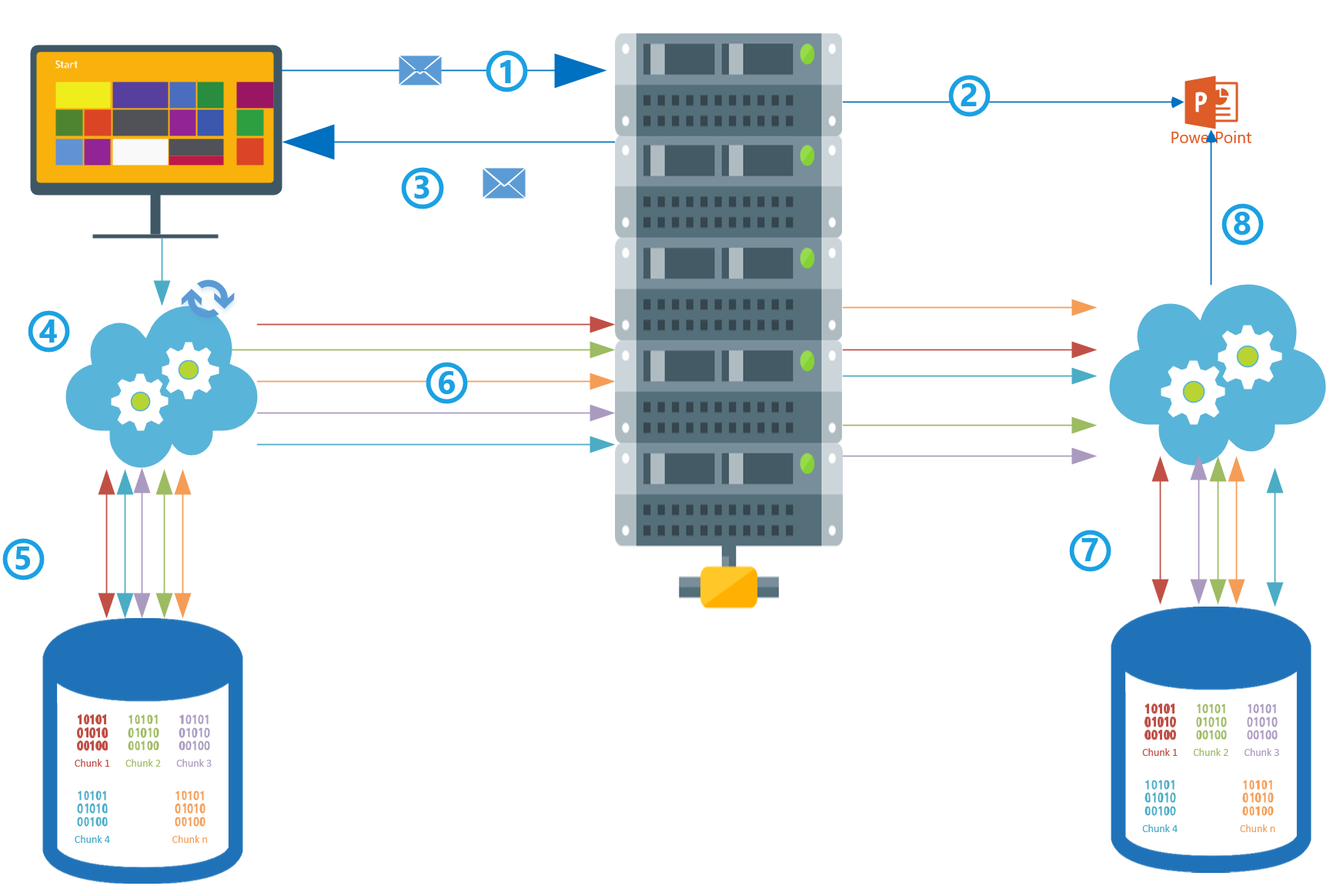


*Architecture at glance - File download diagram*

### File Upload:

The client asks the server for an upload (1). Server will analyze the file, if already present on file share (2), if yes checksum is calculated

Server will reply back (3) with a message



*Architecture at glance - File upload diagram*

Depending on the file checksum client will start or not the chunked upload.

If the checksum is different, of file not present on the server, the client will start chunking file (4) on a local repository 5).

Once the chunk operation is finished, the client will start the upload operation handled by the state machine (6).

The server will receive each chunk and will store it in his repository. (7)

When all chunk is uploaded server will create the file out of chunks (8) and save it on the appropriate location

## Custom Telemetry

On-premise we have the identical approach as in Azure; the difference is that on-premise data are stored in a relational database.

Each action performed on the Client(1) and server are stored on the Application Insight Telemetry (7) we have two different Telemetry category:

1. Application logs, including exceptions
2. Performed actions stored in an anonymous form (e.g., Upload action, project creation action and so on)

Any Business Intelligence can later analyze these data.

# Scaling considerations

Our service architecture is designed and developed the way that it can scale from a very simple installation where all component are developed on a single server. Web APIs, Managed File Transfer Service, Real-Time, Telemetry, Long-term process and even the database.

The solution is also developed with performances in mind, so it means that whatever is doable software side to achieve best performances was done.

When isn’t physically possible anymore to scale software side, the solution can scale at hardware lever, each component can be deployed on a separated physical server.

When scaling requirements go very high, in front of all the architecture, it can be placed a load balancer both physical or software to handle and manage traffic load.