

Beezy

Technical Architecture Overview



Table of Contents

Executive Summary	5
High Level Technical Approach	6
Solution Design Concept	6
Architecture Overview	8
Extensibility and Customization	8
Scalability	9
Performance	10
Interoperability	10
Architecture Models	12
Supported Platforms	12
Installation Models	12
On-premises Classic Architecture	13
Add-in Model Architecture	15
On-premises with Add-ins	16
Cloud Architecture	17
Choosing between On-premises Classic or Add-ins	18
Data Architecture	20
Beezy API	23
REST API	23
GraphQL API	25
Installation Topologies	26
Planning an Installation Topology	26
Single Site Collection	26
SharePoint Site Collection Limits	26
Multiple Site Collections	27
One Site Collection per Beezy Container	29

Multiple Web Applications and Farms	30
Separate Instances of Beezy.....	30
Beezy Networks	31
Security and Access Control.....	33
SharePoint and Beezy Security Architecture	33
Global Administrators.....	35
Security Groups for Features.....	35
Authorship and Access Control.....	35
Other Security Topics.....	36
SharePoint Services Architecture	38
Search Service.....	38
User Profile Synchronization	40
User Profile Service.....	41
Azure AD	42
My Site	42
Managed Metadata Service.....	43
Background Jobs	45
Azure Architecture	47
Azure Hosting.....	47
Azure AD App.....	47
Background Jobs	48
Database Server	48
Monitoring.....	49
Front-end Architecture.....	51
JavaScript Architecture	52
HTML/CSS	53
Third Party Dependencies at Client	54
Front-end Optimization	55
Browser Support.....	56

Configuration and Settings.....	57
Sites Quota	59
Deployment/Upgrade Process	60
On-premises Classic Installation	60
Deployment Components.....	60
Deployment Process	60
Cloud installation	61
Deployment Components.....	61
Deployment Process	61
On-premises Upgrade	62
Cloud Upgrade.....	62
On-premises Add-ins Installation and Upgrade	63
Extensibility.....	64
Branding	64
Classic on-premises Installations.....	65
Add-in Installations	65
API Extensibility.....	65
Multi-language Support	67
Governance.....	68
Logging and Monitoring	68
On-premises Installations	68
Cloud Installations	69
SharePoint Governance.....	69
Backup and Recovery.....	70
On-premises Backup Procedure	70
Cloud Backup Procedure	71
On-premises Full Recovery.....	72
Cloud Recovery Strategy	72
Other Topics.....	74

Data Architecture Diagrams	74
Third Party Components for Server	75
External APIs.....	78
Software Development Lifecycle	79
SharePoint Limits and Boundaries	81

Executive Summary

This document is a high-level description of Beezy's technical architecture. Beezy can be installed in the customer servers (on-premises scenario), in Office 365 (cloud scenario) or a mix of both (hybrid scenario).

The following sections explain the main components that form Beezy. Each section breaks down the smaller components that enable the on-premises and cloud scenarios.

The user experience is fully transparent to what happens in the background. Therefore, the final product is exactly the same regardless the backend infrastructure.

High Level Technical Approach

Solution Design Concept

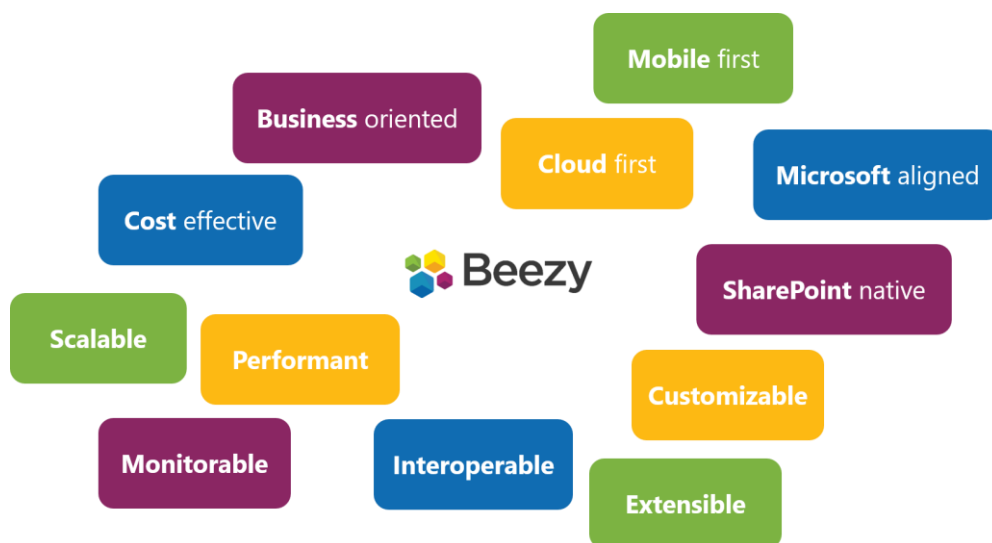
Beezy has been designed as an intelligent workplace solution built on SharePoint and Office 365. Its purpose is to offer advanced collaboration and social features that out-of-the-box doesn't provide (advanced content sharing, advanced collaboration tools, etc.) but also to filter SharePoint's complexity by presenting a simple and intuitive user interface.

Beezy is intended to be as light as possible as a product allowing customers to take full advantage of SharePoint extensibility and customization. This is the reason why Beezy mainly uses standard SharePoint objects or content types like Document Libraries, Calendars or Discussions.

By extensively using the SharePoint Object Model, Beezy can easily adapt to any customization that clients may have done to standard components.

Beezy stores most of its business data in SharePoint lists in order to keep the information inside the platform. This design helps customers reuse standard SharePoint policies like auditing, back-up and restore. Beezy also uses its own database mainly for performance issues and scalability limitations.

Beezy's technical vision is built around the following concepts:



- Business oriented: the company's technical architecture serves the company's core business strategy.
- Cloud first: the majority of our customers are in the cloud or in the process of moving to there, so our architecture must always be aligned with Azure's cloud infrastructure as the challenges it presents are more complex than those of an on-premises deployment.
- Mobile first: our product has been architected assuming that our customers will access the system from virtually anywhere and from a wide range of mobile devices.
- Cost effective: we make an efficient use of the resources to ensure we maximize the capacity and features of the underlying platform.
- Microsoft aligned: to ensure that our customers can leverage the latest features released by Microsoft our product's architecture must go very much aligned with the direction that Microsoft is following in terms of technological strategy.
- SharePoint native: by leveraging all the latest SharePoint components and features we ensure a smooth, fast and reliable experience.
- Scalable: the product must be able to support companies ranging from few hundreds to hundreds of thousands of users seamlessly by making an efficient and smart use of the resources available.
- Performant: as the product grows in features it is crucial to continuously optimize and reduce the load of the pages to ensure fast response times.
- Customizable: one-size-fits-all products fall short especially in large enterprises so our architecture provides easy and reliable ways of customizing and extending the out-of-the-box product.
- Monitorable: the system can be easily monitored at all times to ensure that our customers enjoy minimal downtimes.
- Interoperable: through an extensive and rich API our product can be easily connected with third party systems.
- Extensible: our product has been designed to deliver a rich core set of features while still providing robust and easy ways to enhance, extend and customize the functionality available.

Architecture Overview

Beezy stands on top of the SharePoint platform and provides its own components (a browser front-end layer, a back-end layer, a product API, event receivers for being connected with SharePoint, background jobs) to configure the application interface and logic.

The diagram below shows an overview of Beezy's architecture and its related components.



The Beezy application core (back-end) interacts with a Beezy SQL database and also with SharePoint applications (i.e. Managed Metadata, User Profile, Search Service). The application has a service layer, the product API, which offers all the functionality to the client components (either the Beezy front-end engine, the native mobile apps or other third party custom clients).

Extensibility and Customization

You can provide Beezy with your own organization branding, customizing the look & feel of the Beezy interface components (master pages, CSS style sheets) or providing your existing SharePoint branding infrastructure.

In addition, some user interface capabilities as snippets or praises can be easily configured using the application settings.

From a more advanced customization perspective, you can use the extensibility tools provided by SharePoint Server and Online or Visual Studio to create your own extensions on top of Beezy. For example, you can:

- Override the default behavior of the Beezy UI components by applying your own CSS or JavaScript.
- Create new Web parts or UI customizations that consume and interact with Beezy data (communities, activities, etc.) through the API.
- Create your own site templates to modify the default contents for Beezy communities.
- Use SharePoint feature stapling or feature/event receivers to override the behavior of Beezy communities.

Scalability

The Beezy architecture has been designed to meet any scalability goal our customers may set.

In terms of scalability, Beezy consists of two main components that must be taken into account: the application layer and the SQL database.

For classic on-premises installations, the Beezy application is hosted in the SharePoint Web application context and relies on the scalability schema provided by the platform itself. The application core resides at the Web Front End server tier. Thus, the server topology can be scaled by adding more Web Front Ends to add extra capacity for business logic and request processing. Even after the product deployment has finished, if the number of concurrent users increases customers can meet the demand by deploying more Web Front End servers.

For cloud and add-in installations, the Beezy application is hosted in a separate context, a Web server hosted either in Windows Azure or a local Internet Information Server. Therefore, Beezy can be scaled using standard methodologies and techniques that apply to any other Azure/IIS Web application.

The Beezy SQL database can be scaled using SQL Server clustering mechanisms, such as SQL Server Cluster, SQL Server mirroring or SQL Server Always-On.

Regarding SharePoint scalability limitations, the most important one to consider is the maximum volume recommended for SharePoint content databases but you can overcome this limit by using a multiple site collection architecture, supported by Beezy.

Performance

In order to provide the best user experience, the design of the Beezy architecture has focused especially on some performance topics that needed to be carefully planned.

The data managed by the application is stored in a dedicated SQL Server database instead of using SharePoint as a data repository. SQL data access and processing is faster and more configurable than SharePoint lists. The Beezy database has been monitored, tested and optimized to provide the best performance possible for the queries and operations required by the application and the most common usage patterns.

Most of the typical application requests are normally redundant so Beezy includes different cache mechanisms for data access. Also, as Beezy is part of the SharePoint site schema, it can leverage other platform cache strategies as output cache or blob cache.


All the operations that do not need immediate response are resolved asynchronously, using techniques such as AJAX requests or queues, ensuring the best responsiveness to the end user.

Interoperability

The Beezy API layer offers a simple standardized way of accessing Beezy data regardless of the device type or the technology behind it. All the Beezy core functionality is offered in a complete, self-documented API that does not need further configuration to be used.

Service help page

You are viewing a feed that contains frequently updated content. When you subscribe to a feed, it is added to the Common Feed List. Updated information from the feed is automatically downloaded to your computer and can be viewed in Internet Explorer and other programs. [Learn more about feeds.](#)

 [Subscribe to this feed](#)

IBeezyV2: FollowUser

Today, 25 de setembro de 2016, 16:25:37

UriTemplate	https://dogfood.beezy.net/_vti_bin/beezy/v2/api.svc/Users/Current/Following
Method	POST
Request Format	xml or json
Request Schema	https://dogfood.beezy.net/_vti_bin/beezy/v2/api.svc/help/FollowUser/request/schema
Request Example	https://dogfood.beezy.net/_vti_bin/beezy/v2/api.svc/help/FollowUser/request/example
Response Format	json
Response Schema	https://dogfood.beezy.net/_vti_bin/beezy/v2/api.svc/help/FollowUser/response/schema
Response Example	https://dogfood.beezy.net/_vti_bin/beezy/v2/api.svc/help/FollowUser/response/example
Description	Follow user for the current user

IBeezyV2: UnFollowUser

Today, 25 de setembro de 2016, 16:25:37

UriTemplate	https://dogfood.beezy.net/_vti_bin/beezy/v2/api.svc/Users/Current/Following
Method	DELETE
Request Format	xml or json
Request Schema	https://dogfood.beezy.net/_vti_bin/beezy/v2/api.svc/help/UnFollowUser/request/schema

Displaying 420 / 420

• All 420

Sort by:

▼ Date

Title

Here are some examples of supported scenarios:

- Connecting third party systems with Beezy to create communities or activities connected to events occurred in these systems.
- Creating SharePoint Web parts to add social capabilities to your existing sites.
- Consuming the Beezy API from external systems to show Beezy information embedded on them.
- Creating mobile apps to interact with Beezy data.

Architecture Models

Supported Platforms

Beezy is currently compatible with SharePoint 2013, SharePoint 2016, SharePoint 2019 and SharePoint Online.

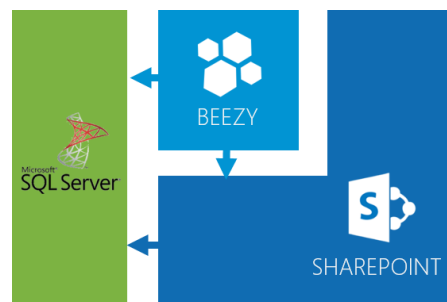
SharePoint Online requires an additional Web hosting platform, Microsoft Azure, for deploying and executing the Beezy components.

Please check the official pre-requisites guide to ensure that all of them are satisfied before installing Beezy.

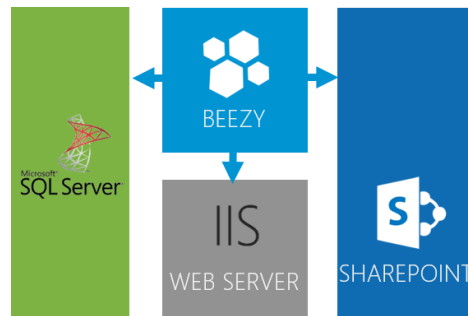
Installation Models

There are 3 installation models, depending on the underlying platform and the SharePoint extensibility architecture used:

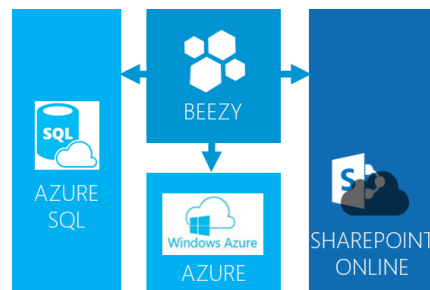
- SharePoint on-premises classic: Beezy installed as a full trust code solution and executed on top of the SharePoint context.



- SharePoint on-premises with add-ins: Beezy installed as a set of on-premises SharePoint Add-ins and executed as a provider-hosted application.



- Office 365 (SharePoint online): Beezy installed as a set of cloud SharePoint Add-ins and executed as a provider-hosted application.

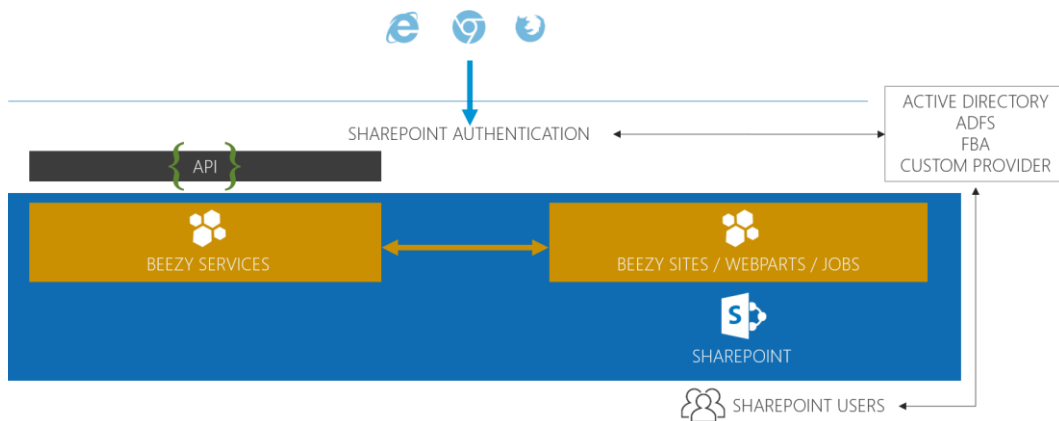


It is also necessary a SQL Server infrastructure for both on-premises and online architectures. For on-premises, the SQL local infrastructure used for SharePoint might be reused. For online, Azure SQL is the proposed database service.

On-premises Classic Architecture

Beezy on-premises can either operate with server side model (the classic approach of full trust code and farm solutions) or add-in model (using high-trust SharePoint add-ins). The first installation model is named Beezy classic on-premises installation.

This is Beezy's overall architecture according to the classic model:



Beezy's components are deployed using SharePoint solution packages (WSP's). These components are:

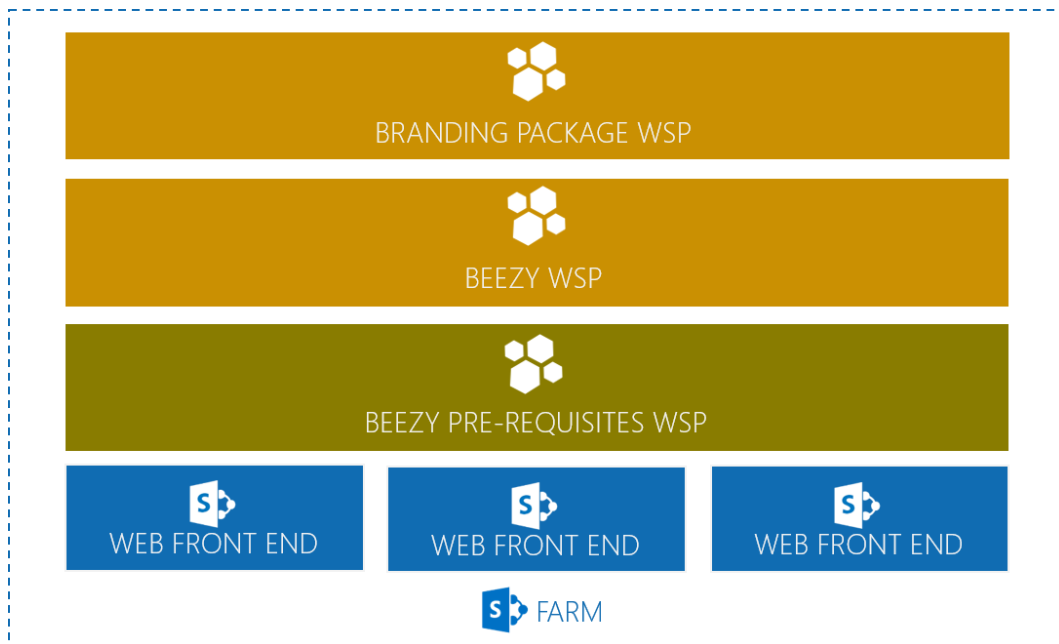
- DLL's implementing the application logic
- An API which offers the application functionality and operations
- Artifacts for SharePoint synchronization like timer jobs or event receivers
- User interface components (master pages, CSS, JavaScript)

The SharePoint packages involved in a Beezy setup in server model include:

- A Beezy WSP containing the components described above.
- A pre-requisites WSP containing the necessary third party DLL's.
- An optional branding WSP containing the customized branding files (master pages, CSS, JavaScript) for a specific client or project.

These three packages are deployed in each Web application where Beezy is needed.

The following figure shows the overall structure for WSP deployment in Beezy:



The main tiers in the application are Client (Web browser) and Server, the separation is the product API. The Server tier scales through the out-of-the-box SharePoint Web Front-End scalability schema.

Add-in Model Architecture

The idea behind the add-in model architecture is to execute every single piece of Beezy logic outside the SharePoint context to avoid any unnecessary resource consumption in the SharePoint farm. The required interaction with the SharePoint components is done using the Client-Side Object Model (CSOM), which is a safer interface to consume SharePoint artifacts and resources. Most Server Object Model operations are also available in the CSOM, but there are also some significant differences between the classic and add-in models.

In the add-in model, Beezy is not distributed as farm solution packages, but as a provider-hosted application (add-in). The Beezy add-in provisions the components needed on the SharePoint side.

Instead of site templates, the add-ins are responsible for automating the deployment and configuration of SharePoint artifacts (lists, content types, pages, client Web parts etc.). Client Web parts (also known as add-in parts) render the content from the Beezy application. The separation between environments does not affect the user experience. Users can see

embedded Beezy content in their SharePoint sites, just like in the classic model deployment.

In the classic model, event receivers are attached to lists or content types and run in the SharePoint context. In the add-in model, they run as remote event receivers in the Beezy application context.

In the classic model, timer jobs are deployed as part of the SharePoint deployment. In the add-in model, they are converted to scheduled jobs, which are managed and executed by the Beezy application itself. This implies that administrators cannot have a consolidated view of all the Beezy and SharePoint jobs. They have to manage them separately through different interfaces.

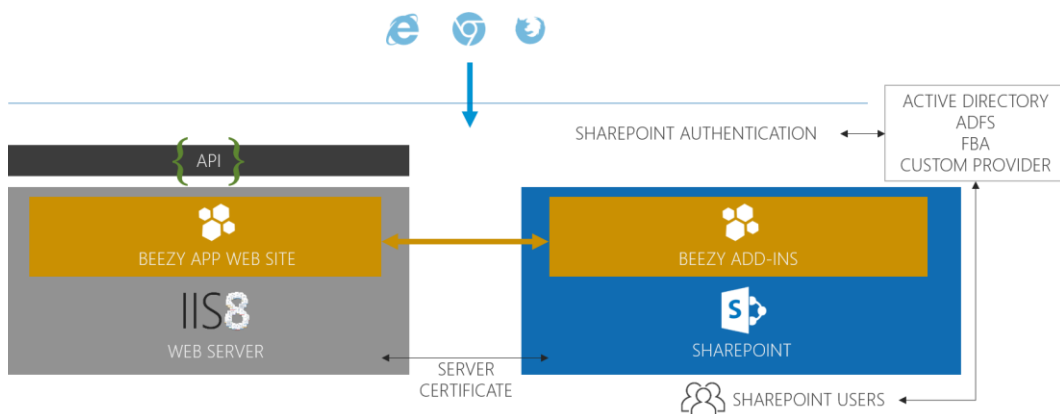
The Beezy API and the Beezy SQL repository are also hosted outside of SharePoint. This means that Beezy needs a specific capacity plan, different from SharePoint. As Beezy is hosted in an independent Web server topology, it does not depend on SharePoint infrastructure but its own Web and SQL sizing and scaling.

As Beezy is a separate ASP.NET Web application, the add-in architecture fits in both on premises and cloud scenarios. In an on-premises scenario, Beezy needs a separate local IIS/ASP.NET/SQL Server infrastructure with connectivity to the SharePoint farm to consume the SharePoint resources through Client-Side Object Model (CSOM).

In a cloud scenario, the ASP.NET Web application is deployed to Azure and the Beezy SQL repository to Azure SQL Services. The Beezy add-in is registered to an existing SharePoint online instance, so the SharePoint resources are made available to the Beezy App through the CSOM.

On-premises with Add-ins

This is Beezy's overall architecture according to the on-premises add-in model:



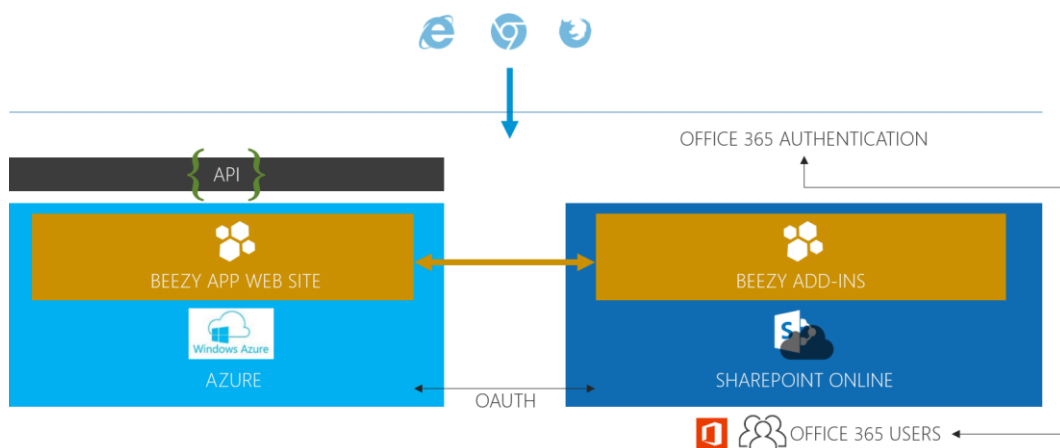
Beezy is provided as a high-trust SharePoint add-in, which is registered in the SharePoint side, but all the Beezy application logic is executed in a separate Internet Information Server.

The user authentication relies on the standard SharePoint authentication providers configured in the platform. However, the authentication between the Beezy add-ins and SharePoint is based in server certificates that provide the trust relationship between both parts.

It is important to check the pre-requisites guide about Beezy on-premises with add-ins to understand the requirements and implications of configuring high-trust SharePoint Add-ins in your environment.

Cloud Architecture

This is Beezy's overall architecture according to the cloud add-in model:



Beezy is provided as a low-trust SharePoint add-in, which is registered in the SharePoint side, but all the Beezy application logic is executed in a separate Azure web site.

The user authentication relies on the standard Office 365 platform authentication. However, the authentication between the Beezy add-ins and SharePoint is based in OAuth, according to the standard design of low-trust add-ins.

Choosing between On-premises Classic or Add-ins

Beezy supports two different installation models for on-premises deployments: classic (full trust code solutions) and high-trusted add-ins. The entire set of product features and scenarios are available in both models so the reasons to choose one or the other depend solely on technical and strategic reasons.

First of all, installing an on-premises classic Beezy is a very simple process compared to getting a Beezy on-premises with add-ins ready. High-trust add-ins require a considerable amount of preparation and configuration steps in the environment. These steps are not only required for Beezy add-ins but for every provider-hosted add-in the customer wants to install in the SharePoint local farm.

This means that if Beezy is the first SharePoint add-in that the customer is installing in the farm, the Beezy installation will trigger an initial configuration process and a set of pre-requisites that will impact in the overall setup time. If Beezy is installed in an environment where high-trust add-ins are already common and there are well-known policies and procedures to implement these kind of installations, Beezy's setup will be a much more lightweight process.

Beezy's general recommendation for deciding when to install each model is considering two main factors: the maturity of the environment with regards to high-trust add-ins and the strategic requirements that enforce the use of add-ins in a particular environment.

Another deciding factor that customers might want to take in to account is the company's technological long term strategy. The existence of a technical roadmap for moving to the cloud might be a good reason to adopt the add-in architecture early on even in an on-premises deployment. In addition, for hybrid scenarios it makes much more sense to have a homogeneous architecture between on-premises and cloud deployments.

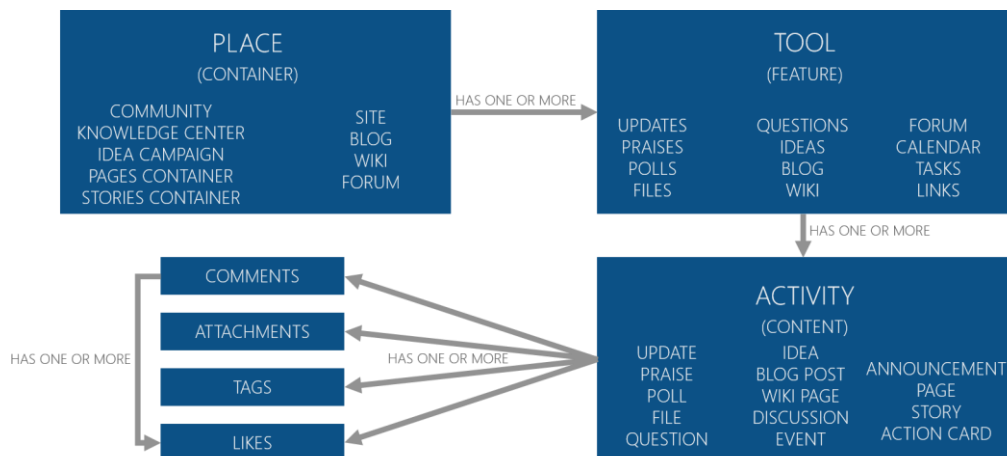
On a final note, Beezy always recommends to choose the most straightforward approach and follow the classic installation model when there are no specific enforcement policies that require SharePoint add-ins. This will avoid overcomplicating the first Beezy installation process and the customer can always easily migrate from a classic installation to an add-in installation model.

Data Architecture

The architecture for implementing an intelligent workplace solution based in Beezy is a mixed approach of SharePoint and Beezy itself. The main objectives behind this architecture are:

- High integration of SharePoint and Beezy.
- All the collaboration data is stored in SharePoint repositories.
- All the social data is stored in the Beezy repository.
- Beezy as an intelligent workplace built on SharePoint, offers an integrated user interface combining collaboration and social data
- In order to achieve full integration and performance, some collaboration data needs to be redundantly stored in the Beezy repository.

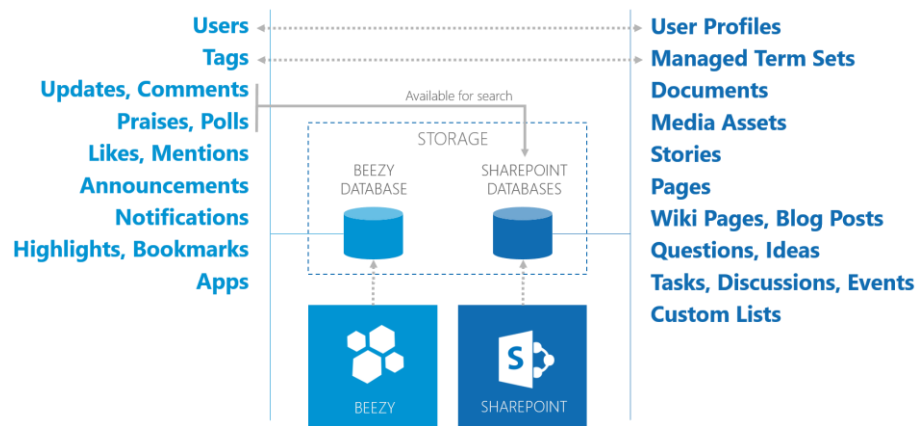
The following diagram represents the main data concepts and relationships in the product:



Places are the main container where information resides in Beezy. Each place might have one or more tools. Some containers (like communities or sites) allow the configuration of these tools, but the rest of containers have a set of predefined tools or just one of them.

The generic name for the any type of content in Beezy is *activity*. There are many types of activities, but all some them (with some specific exceptions) might have comments, attachments, tags and likes.

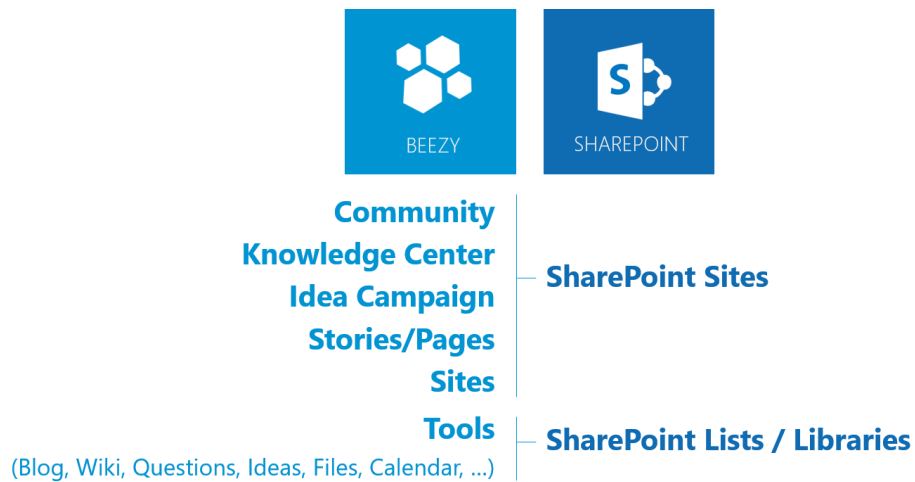
Beezy's data architecture can be layered using a global data mapping between SharePoint and Beezy. The following diagram shows where the information resides for each functionality and data item in the application:



As the diagram shows, the most important data objects in the application are stored in both systems:

- Users are stored as SharePoint user profiles, but Beezy also stores these profiles with extended information.
- Places are created as SharePoint sites, and members of each place are configured with SharePoint site permissions to leverage the platform security.
- Tags use SharePoint Managed Metadata term sets to leverage faceting in Enterprise Search.
- The activities are always stored in SharePoint internal lists, in order to be crawled by Enterprise Search.
- Tools like files, questions, blogs, wikis, forums, tasks, etc. are SharePoint-centric. However, Beezy needs to store certain data from those tools and their items, as a mirrored repository, in order to build a consistent and performant interface that integrates each tool with the content around it.
- Other content only exists in Beezy's scope because there is no direct translation to a pure collaboration platform like SharePoint: likes, comments, mentions, notifications, highlights, bookmarks, etc. is the core social data managed by Beezy.
- Features like user follows, links or feeds have similar SharePoint features but the proposed solution aims to implement them through the Beezy engine due to strategic reasons that will be described in each section.

The following diagram describes how places (communities, knowledge centers, idea campaigns, etc.) map to SharePoint sites and how tools map to particular SharePoint lists and document libraries:



Please check the [Data Architecture Diagrams](#) section at the end of the document in order to get more details about this topic.

Beezy API

The Beezy API is the programming interface to the product. You can use the API to integrate your application with Beezy consuming or feeding the information contained in our database. The API offers a simple standardized way of accessing Beezy's data regardless of the device type or the technology behind it.

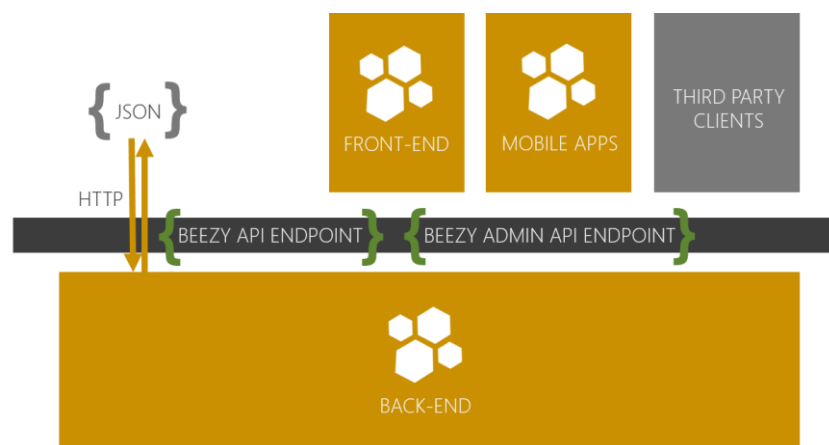
The API of the product has two main different endpoints: the REST API and the GraphQL API. Currently the REST API covers most of the application functionality and the GraphQL endpoint covers the following functionalities: sites, pages and apps.

The Beezy front-end engine itself uses the API to build the Beezy UI and provide the available operations. Therefore, from a usage point of view, all Beezy components are using the API to perform their operations. In other words, if the product front-end can cover a certain use case, most likely a third party can do that same through the API.

REST API

The Beezy REST API uses HTTP as the connection protocol and JSON for message encoding.

It has two different endpoints: the main API endpoint for providing all the end-user operations and the administration API endpoint for providing the administrative operations.



Internally, the API is built using ASP.NET/WCF technology and it receives and sends JSON formatted messages to the client, using the standard HTTP verbs GET, POST, PUT and DELETE. The messages are returned using a standard model based on DTO's (Data Transfer Objects).

```

▼ [{"Attachments": [{"Created": "/Date(1474539294000+0000)/",...}], Created: "/Date(1474539294000+0000)/",...}]
▼ 0: {Attachments: [{"Created": "/Date(1474539294000+0000)/",...}], Created: "/Date(1474539294000+0000)/",...}
  ▶ Attachments: [{"Created": "/Date(1474539294000+0000)/",...}]
    Created: "/Date(1474539294000+0000)/"
    CurrentUserLikedActivity: false
    Device: ""
  ▶ Group: {AppVersion: null, Color: "32cb12", Description: null, GroupId: 48, IsActive: true, IsModerated: false,...}
  ▶ Hashtags: [{Id: 1174, Name: "Add a tag..."}]
    Highlighted: false
    Id: 3634
    Mentions: []
    Message: ""
    MetadataProperties: []
  ▶ Owner: {AboutMe: "", BackgroundPictureOffsetX: 0, BackgroundPictureOffsetY: 0, BackgroundPictureUrl: null,...}
    PreviewImageUrl: "/groups/48/MediaGallery/enric_carrión-150x150.png"
    RelatedItem: null
    TotalOfComments: 0
    TotalOfLikes: 0
    TotalOfTags: 1
    Type: "document"
  ..
  ..

```

The API reference can be found browsing the following URL's in any Beezy installation.

- Classic on-premises
<http://<site collection>/vti bin/Beezy/v2/Api.svc/help>
<http://<site collection>/vti bin/Beezy/v2/Administration.svc/help>
- Add-ins
<http://<beezy addin website>/BeezyApi.svc/help>
<http://<beezy addin website>/BeezyAdminApi.svc/help>

This is an ATOM feed that can be easily opened by any available tool.

For installations with add-ins, please provide the following parameters to any API call:

- SPHostUrl: SharePoint site URL where to apply the API call (context)
- AppName: Beezy
- SPAppToken (only in cloud): SharePoint online access token for authenticating the user

There are several common ways how to call the API:

- PowerShell (Invoke-RestMethod)
- Beezy.Client DLL (provided with the installation package)
- Javascript Client API (AJAX)

GraphQL API

The Beezy [GraphQL](#) API uses HTTP as the connection protocol, JSON for message encoding and the GraphQL query language for message syntax.

It has one single endpoint for providing all the end-user operations about sites, pages and apps.

Internally, the API is built using .NET/GraphQL technology and it receives and sends JSON formatted messages to the client, through GraphQL queries and mutations.

The API reference can be found browsing the following URL's in any Beezy installation.

- Classic on-premises
http://<site_collection>/layouts/15/Spenta.Beezy/GraphQL.aspx
- Add-ins
http://<beezy_addin_website>/GraphQL

This reference is a Web interface called [graphiQL](#), which allows navigating through the API schema, calls and performing operations directly on the environment.

For installations with add-ins, please provide the following parameters to any API call:

- SPHostUrl: SharePoint site URL where to apply the API call (context)
- AppName: Beezy
- SPAppToken (only in cloud): SharePoint online access token for authenticating the user

Installation Topologies

Planning an Installation Topology

Before installing Beezy in any productive environment, an initial assessment of the requirements must be performed in order to determine an installation topology that will allow the system to scale properly. This chapter describes the most frequent topologies that customers can use to deploy the product. For other advanced topologies or requirements please contact Beezy.

Single Site Collection

This topology is the simplest and it is ideal for proof of concept scenarios, testing and reduced implementations. All Beezy installations can start being a single site collection scenario and evolve to multiple site collections later on.

Each place has its own subsite. They are all hosted in a flat structure as subsites of the '/groups' subsite.



In a Beezy installation with a single site collection it is very important to monitor and ensure that the site collection is always under the supported SharePoint limitations (see section below). If at some point the data growth makes the site collection reach these limits, it is highly recommended to scale up the product by adding new site collections. See the sections below to find more about how to solve this scenario.

SharePoint Site Collection Limits

There are two key platform limits in SharePoint that might affect the scalability and distribution of sites within site collections.

SharePoint on-premises:

<https://technet.microsoft.com/en-us/library/cc262787.aspx>

Staying below 2,000 subsites per site collection is strongly recommended

Content database size (general usage scenarios) 200 GB per content database

Content database size (all usage scenarios) 4 TB per content database

SharePoint Online:

<https://support.office.com/en-au/article/SharePoint-Online-software-boundaries-and-limits-8f34ff47-b749-408b-abc0-b605e1f6d498>

Subsites Up to 2,000 subsites per site collection

Site collection storage limit Up to 1 TB per site collection

The first limitation (number of subsites) enforces to host a maximum of 2,000 sites in a single site collection, which is, a maximum of 2,000 places. Note that tools inside places do not count, as they are not hosted in a separate subsite. If a customer plans to create a number of places that exceeds the SharePoint recommended limits, please plan a multiple site collection topology as described in the following sections.

The second limitation (storage) enforces to always keep the site collection total size under the recommended amount, depending on your current SharePoint infrastructure. Everything considered as primary content in Beezy (files, images, blog posts, wiki pages, questions, etc.) is stored in the SharePoint content database, so it must be taken into account when computing the total content size needed. If a customer plans to host a total amount of data that exceeds the SharePoint recommended limits, please plan a multiple site collection topology as described in the following section.

Multiple Site Collections

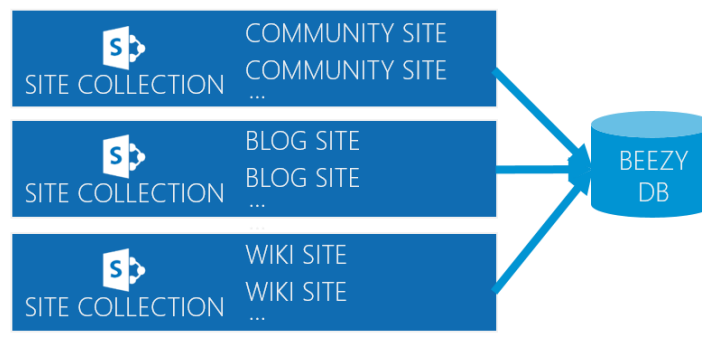
Beezy supports multiple site collections in a single installation. When having multiple site collections, one of them will be selected as the main (default) site collection, from a navigation perspective. The same or another site collection will be selected for creating new sites.



This way, every time that users use the option “Create a new community” (or any other type of place), they will be redirected to the site collection configured to create new communities by default. However, all site collections contain all the Beezy artifacts, which means that, at any point, administrators can create communities in any site collection.

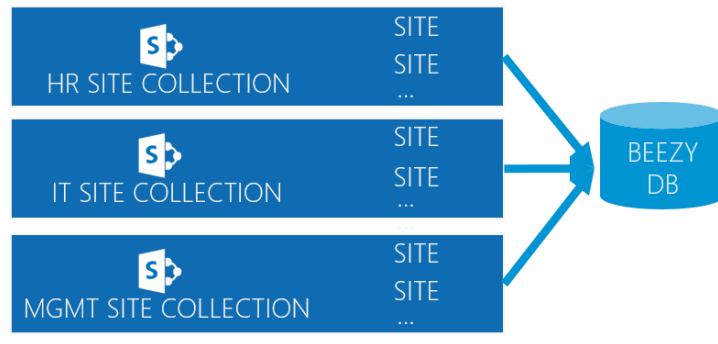
It is important to note that the content from the different site collections is shared across the application, so there is only a single consolidated view (the global newsfeed) of the posts, comments, questions, etc., no matter how many site collections are hosting the communities in the background.

When planning an installation without specific requirements, Beezy’s default proposal is to separate each type of place (community, stand-alone blog, stand-alone wiki, stand-alone forum, knowledge center, idea campaign, etc.) in different site collection types. For every type, the sites will be distributed in one or more site collections depending on the number of sites to host.



Customers can define different site collections for creating communities, blogs, wikis, forums, idea campaigns, knowledge centers, sites, etc.

When defining a topology with multiple site collections, Beezy can also adapt to any existing organizational structure based on business units, divisions, regions, building a separate site collection for each purpose.



One Site Collection per Beezy Container

Beezy also supports the scenario of creating a different site collection for each Beezy place (community, blog, wiki, etc.). However, the main implication to consider when building this scenario is that the product does not provide any out-of-the-box interface to achieve this.

When planning a topology where every community is a new site collection, the customer's IT department needs to be involved in the process of defining the provisioning mechanism of a new community. This process can be automated using scripting (i.e. PowerShell) or implemented through an alternative user interface. Normally this scenario is associated to site collections with a certain degree of customization which can be included in the provisioning mechanism.

Beezy does not provide self-service creation of site collections (self-service creation is for subsites), but it offers SharePoint features for on-premises and SharePoint Add-ins for cloud to automate the provisioning of a new site collection and convert it to Beezy community, blog, wiki, etc.

During the planning phase of each project, Beezy can offer extensive guidance to help building the provisioning mechanism.

The default recommendation for these scenarios is to create a separate Beezy site collection that acts as the Beezy main hub, so the settings management is performed through this hub.

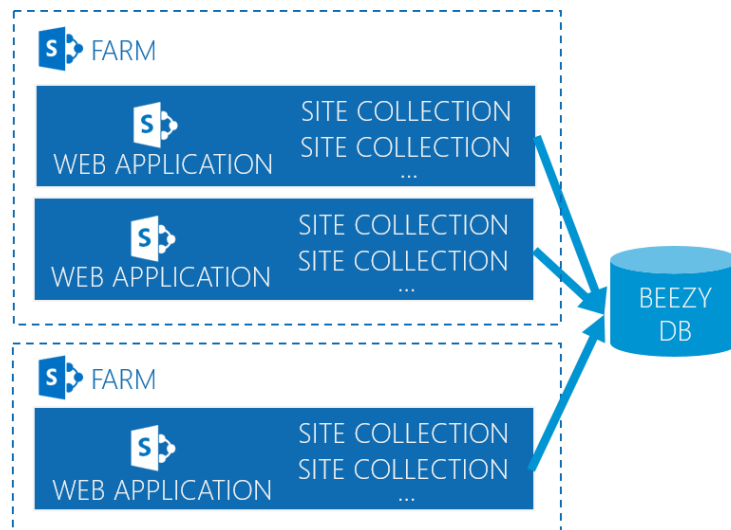
Multiple Web Applications and Farms

Beezy supports installations across multiple Web applications and SharePoint farms but there are some remarks around these scenarios.

As the Beezy database acts as a hub for all site collections where the product is installed, customers can connect site collections belonging to different Web applications or farms to the same database. Most of the aggregated queries and functionalities will use the database to retrieve the actual Beezy content, so for most product features there is no impact.

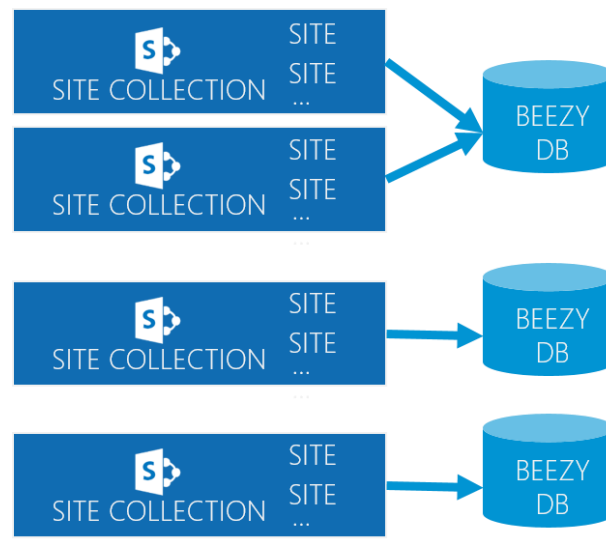
However, specific use cases like publishing SharePoint content from one Web application to another (i.e. sharing a Wiki page from the Beezy start page browsed from a Web application to a community hosted in a different Web application) are not supported by the out-of-the-box Beezy.

In order to consider scenarios with multiple farms and Web applications, Beezy needs to perform an assessment of the scenario and plan accordingly the possible customizations involved that affect the user interface or additional integrations with the API.



Separate Instances of Beezy

Customers can have more than one Beezy instance in a single environment. This means, separate Beezy spaces, isolated one from another. Each instance will have its own Beezy database.



It is important to note that for full trust scenarios in on-premises, the Beezy core artifacts (DLL's, resource files, etc.) will be shared between all instances. For Add-ins scenarios, the Beezy add-in will also be shared between all instances.

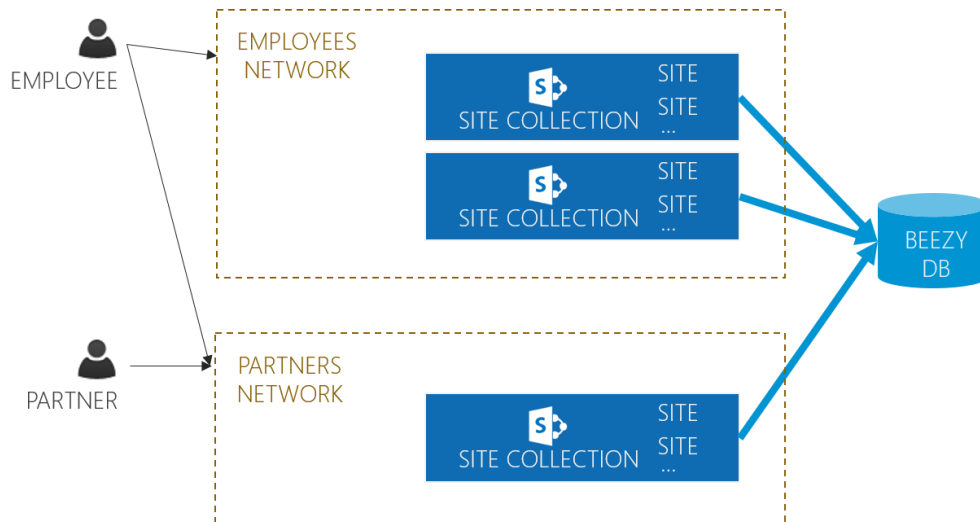
Beezy Networks

The concept of Networks is a Beezy product feature that allows configuring advanced topologies within a single Beezy installation. This feature is built as a framework in the product to help accommodate different scenarios but it does not cover all possible cases and combinations. The example below illustrates the most common scenario that can be accomplished with Beezy networks.

A customer might want to enable Beezy not only for corporate users (employees) but also partners or other groups of external users. With a regular Beezy topology, if employees and partners access the same Beezy instance both will have visibility of the same public communities, which might not be desirable in many cases.

With networks, Beezy allows to define a new area for partners within the same instance. For this area (network), the customer defines which users (or groups of users) can access it. Then, the customer can create public content (communities, blogs, wikis, etc.) but targeted for partners. Thus, partners can access this specific network but not the default (employee) network. On the other hand, employees can access both networks because they have been granted access to both of them. Employees and partners

can interact together in those communities created in the partners' network.



Beezy allows the possibility to create several networks in a single instance. Each network consists of one or more site collections; one of them is the main (root) site collection of the network, so each network has its own start page. Each network can also have its own branding. However, all networks share the same Beezy global configuration settings.

There is an even more advanced scenario that can be built with networks: if a customer enables two or more networks in the same site collection, it is possible to change the network of a community after it has been created. Therefore, a community for employees could be at some point promoted to a public-facing community to interact with partners also. However, this scenario is not recommended by default as it can raise security concerns related to community owners handling irresponsibly sensitive content and making it accessible to non-employees.

Security and Access Control

SharePoint and Beezy Security Architecture

SharePoint offers a complete and robust security framework that allows configuring and customizing different groups and roles to different levels of the data architecture: site collections, sites, lists, etc. All user interaction with SharePoint elements must be secured with the proper SharePoint permission level, it is not enough to cut access to SharePoint content using the interface, the application also needs to prevent content access through direct URL's or API's.

There are two main levels to secure: the site collection and the sites for each place. At the site collection level, the standard groups in SharePoint (Owners, Members, Visitors) are not used. Instead, the product installation creates another group called Beezy Network Associates to manage access permissions to Beezy.

The Beezy Network Associates group is used to ensure availability for all the users that have access granted to the application. This group will include all people that needs to access the application. Typically, they will be included through one or more Active Directory groups. The global permission for this group will be Read.

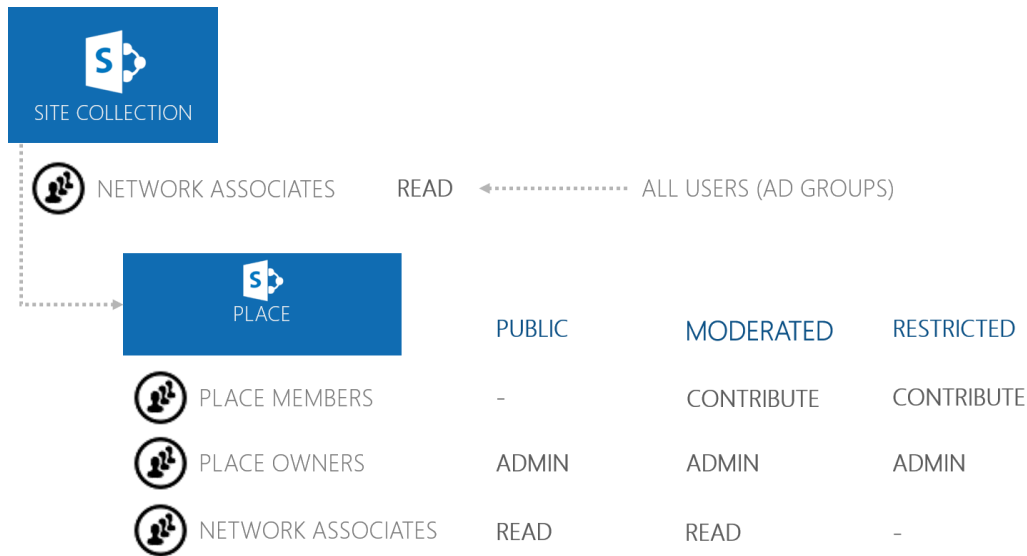
Every site collection will need the same users in this group. In order to maintain this synchronization, the network management provided by Beezy helps to synchronize the users across all the site collections.

Please bear in mind that if a customer grants a new user access to Beezy by adding them to this group, as soon as this person visits any Beezy site collection for the first time, the corresponding user will be created.

At the place site level, there are two SharePoint groups for each place: Place Members and Place Owners. The first one is used to grant generic contribution permissions for the place and the second one is used to grant administration permissions.

There are three privacy levels in places: public (all users can contribute), moderate (all users can access but only place members can contribute) and private (only place members can access).

This diagram is a summary of this permission configuration at the SharePoint side:



The Beezy application propagates automatically the place owners and members specified in the Beezy UI to each corresponding SharePoint group. In fact, this is the supported way to manage place owners and members: doing it through the Beezy user interface and let Beezy synchronize accordingly the SharePoint groups.

Apart from the SharePoint permission schema, there is also a security layer at the Beezy application side responsible to apply security checking when accessing to social data. In order to do that, place members and owners are also stored in the Beezy database. In addition, Beezy stores additional role information, i.e. the place followers.

Having a security schema in the Beezy database provides a performant way to retrieve and manage social data without needing round trips to SharePoint. Every time a call is made to the Beezy API, the application checks the database using the credentials of the connected user filtering the social data by the list of places the user has rights to see. Using this filter, Beezy can ensure that the operation is allowed for the given user.

Global Administrators

In every Beezy installation, some users have to be assigned as global administrators. In order to grant global administration permissions to a user in a Beezy installation, this user needs to fulfill the following requirements:

- Be a SharePoint **site collection administrator** in all site collections where Beezy is installed.
- Be added to the “**Administrator accounts**” field of the Beezy global settings page.

The reason for having this duality is the same performance reason mentioned in the section above: avoid roundtrips to SharePoint.

Security Groups for Features

Apart from the main groups described above, Beezy has the following internal security groups to grant specific permissions to certain roles of the application:

<i>Beezy Community Creators</i>	Members of this group can create new communities. If community approval is enabled, they will only be able to create private communities and requests for new public or discoverable ones.
<i>Beezy Knowledge Center Creators</i>	Members of this group can create new knowledge centers.
<i>Beezy Ideas Campaign Creators</i>	Members of this group can create new idea campaigns.

Authorship and Access Control

For most of the tools of a certain place (blogs, files, forums, etc.), there is an important requirement regarding access control, which is, users (members) cannot edit or delete items that they did not create.

SharePoint has an option in custom lists (not page or document libraries) to allow users to edit their own items. This option is not handled as a real permission configuration. When you configure this option, users will still see the "Edit" option, access the edit form and only when they click Submit, SharePoint will return a generic "Access denied" error page. This is how SharePoint works out-of-the-box and there is not much Beezy can do to improve that.

Beezy has implemented the following two customizations to improve this scenario:

- In the Beezy interface, check if the user is author of the item/document and show the "Edit" option only in this case.
- For page/document libraries, associate SharePoint event receivers that will prevent edition the same way that SharePoint out-of-the-box is doing for custom lists (i.e. if you are trying to edit or delete a community file that you did not upload, you will get an error).

It is important to note that we have discarded the approach of breaking permission inheritance at item/document level due to the limitation of 5,000 unique permissions in a list as stated in the following link. This is not a hard limit but it can affect performance and it should not be exceeded.

<https://technet.microsoft.com/en-us/library/cc262787%28v=office.15%29.aspx>

Other Security Topics

- Data validation: at server-side, data validation is performed at all layers (service, business logic, data access). At client-side data validation is also performed using client frameworks. Conditional fields, expandable fields or calculations are performed using a rich JavaScript layer that uses views, forms and components to wrap all this client logic.
All transactions are resolved in a single request or operation, without the possibility to inject or alter data.
- API access: the Beezy API access is secured for each method. The API authentication is using the authentication provider configured in the web infrastructure that hosts the API services. In addition, each method applies the authorization policies based on the roles and groups described in the previous sections. For example, all

operations in the Beezy administration REST API endpoint will only authorize global administrators.

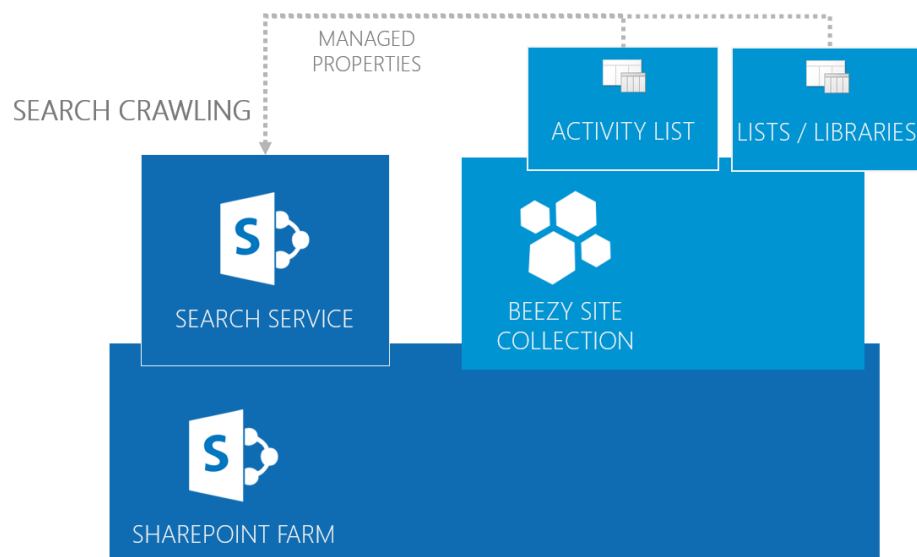
SharePoint Services Architecture

Search Service

Beezy users are also SharePoint users and they expect Beezy search results to be integrated in the out of the box SharePoint search results. Beezy leverages the SharePoint Enterprise Search infrastructure in order to make this integration possible.

During the Beezy installation, some SharePoint search managed properties are created and configured, in order to make certain data available for the SharePoint search crawl processes.

These managed properties point to specific SharePoint columns on lists and libraries stored in the Beezy site collection. Some content in Beezy is indexed from their own document libraries (like stories or pages), must most content is indexed from a hidden list that exists in every Beezy site collection, called the Activity List.



The Activity List is created in the root site of every site collection and it contains information related to places, activities and comments. This information is also stored in the Beezy database but Beezy needs to replicate it as a shadow copy in SharePoint.

As Beezy maps activities and comments to list items it is straightforward to have this information fully integrated.

The Activities List contains a folder for every place in the site collection. This folder has the same permissions as the corresponding SharePoint site. Inside this folder, there is a subfolder for every activity created in the place and, inside this folder, there is a Comments folder containing all comments created for this particular activity (as SharePoint list items).

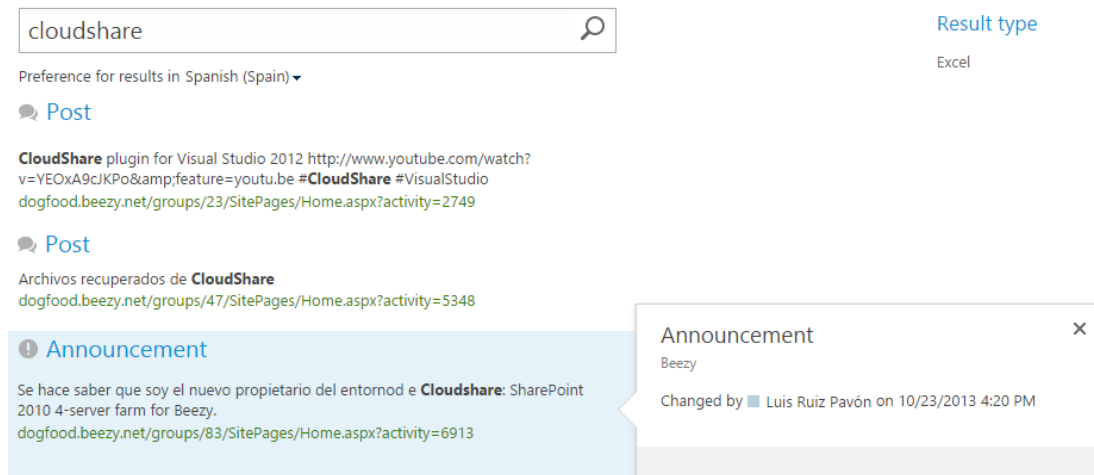


About search results rendering, Beezy makes use of result types and item display templates.

Item display templates are used to control how you want information in the body of the Search Results Web part to be displayed. SharePoint comes with a set of default display templates available but, since Beezy is leveraging SharePoint search using its own set of SharePoint content types and managed properties, Beezy also deploys its own display template to the Master Page Gallery, inside Display Templates.

In order to display search results differently, search results are sorted into different result types. Beezy also deploys its own search result type which links to the Beezy display template.

The Beezy display template renders the result in a different way, depending if it is an update, a question, an event, a praise, etc.



The Beezy item display templates can be reused in SharePoint search centers outside the Beezy site collections, so Beezy results can be rendered in a global company search, together with results coming from many other sources.

User Profile Synchronization

The users in Beezy need to be synchronized with an external source. There are two different sources when planning user profile synchronization in Beezy: getting the profiles from SharePoint User Profile Service or from Azure Active Directory (only in cloud installations). Customers can choose one or another.

Accessing any of these sources is slow and high consuming, so Beezy uses a copy of the needed information in the database. A user profile sync job is executed daily in order to:

- Keep some user fields (Full name, E-mail, About me, Personal Site, SIP Address...) synchronized.
- Add automatically new users to the Beezy database when they are created in the external source.
- Mark users as inactive in the Beezy database when they are removed from the external source.
- Synchronize custom user fields so Beezy can display them as additional user profile information.
- Customers have the ability to filter which user profiles are going to be imported, depending on the value of a configurable user profile property.

In order to ensure that profile information is updated and fresh, Beezy provides different mechanisms:

- User profile incremental synchronization: performed on a daily basis, imports changes occurred in the external source since the last execution of the job.
- Individual on-demand synchronization: on some special situations, administrators might need to synchronize the information of an individual user from the external source to Beezy, through the Beezy administration UI.
- Batch on-demand synchronization: when configuring an environment for the first time or when a significant amount of new users need to be added/synchronized, Beezy provides PowerShell scripts to perform batch synchronization based on a CSV list, an AD group or the whole external source directory.

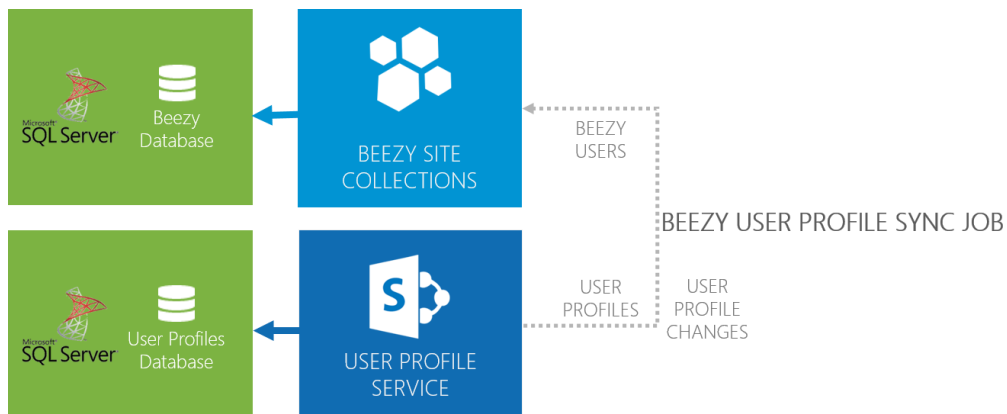
The following properties are synchronized during the job execution:

- Name (Full Name)
- Picture URL
- Work Phone
- Sip Address
- Department
- Job Title
- Office
- Email
- About Me

Each customer can configure which of these properties are synchronized from SharePoint to Beezy. In addition, as Beezy allows users to modify themselves properties like About Me or Job Title, there is the possibility to configure the synchronization back of these values to SharePoint.

User Profile Service

User information is stored in the SharePoint User Profile database and accessed through the SharePoint User Profile Service Application.

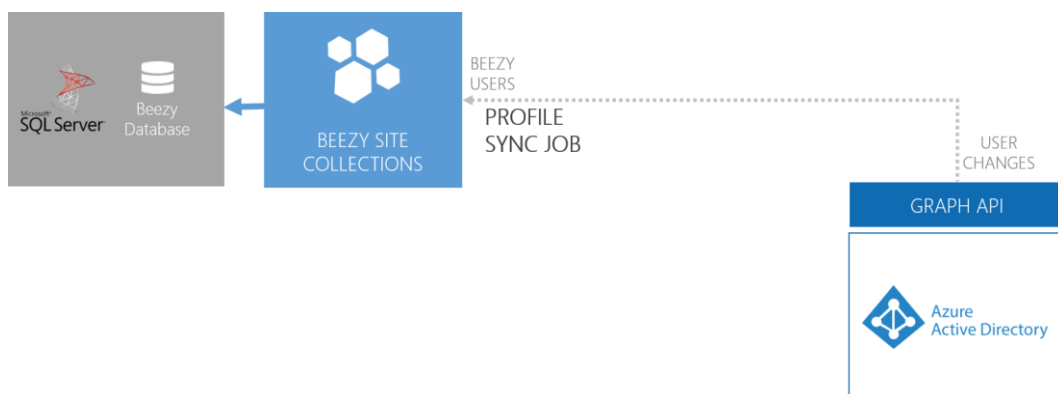


In the on-premises classic model, Beezy relies on the UserProfileChange API (<http://sharepointpromag.com/sharepoint-2010/monitor-sharepoint-user-profile-changes>) to query and process only the changes occurred during the day before, avoiding a full scan of all the user profiles.

In the add-ins model, Beezy uses the SharePoint User Profile Change Web service instead (<http://msdn.microsoft.com/en-us/library/aa981239%28v=office.12%29.aspx>) to retrieve the changes.

Azure AD

User information is stored in the Azure Active Directory of the organization and accessed through the Microsoft Graph API.



In order to consume the Graph API, Beezy requires an Azure AD app, as explained in the [Azure AD App](#) section.

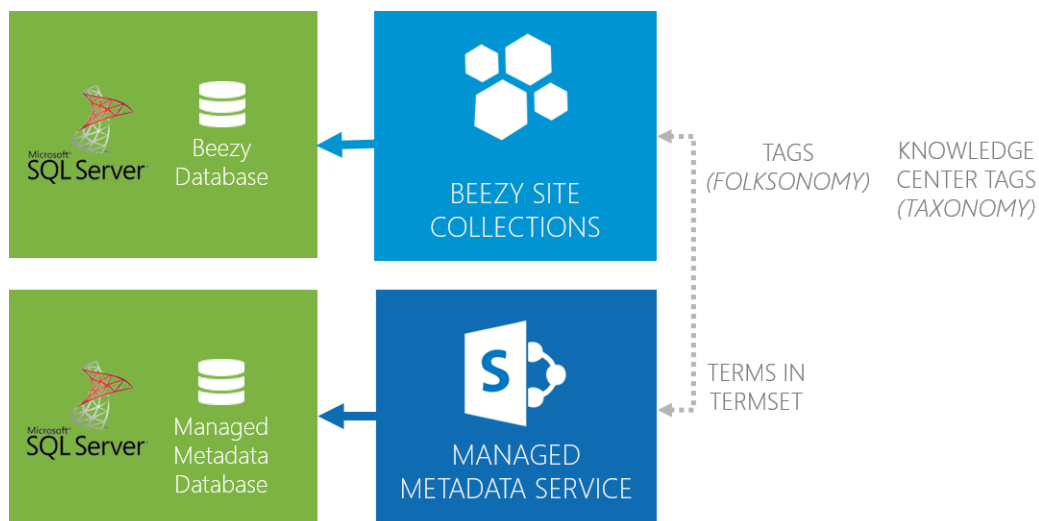
My Site

The SharePoint My Site is not a mandatory service for Beezy. Beezy can be installed and can handle SharePoint user profiles without having My Sites created for them.

Managed Metadata Service

Beezy has the following integration points with the Metadata Management Service:

- When deployed, the application creates a group "Beezy" and an empty term set "Beezy" in the default term store.
- Every time someone posts an activity with one or more tags or someone creates a community with associated tags, those tags will be added to the "Beezy" term set, all at the same level. These tags are the Beezy folksonomy, created by the users themselves.
- The activity SharePoint list (secondary storage for activities) stores the associated terms for each activity, using a Managed Metadata column based on the "Beezy" term set. Thus, this associated column can be indexed and searched by the SharePoint Enterprise search.
- Knowledge centers, a specific type of place in Beezy, uses rigid taxonomy instead of folksonomy. The taxonomy is synchronized from the existing SharePoint term sets and terms in the Managed Metadata Service. Then, users can use this taxonomy to tag content in the knowledge center.



However, there is an important platform limitation regarding SharePoint Managed Metadata. According to the Software boundaries and limits for SharePoint on-premises and online (<http://technet.microsoft.com/en->

[us/library/cc262787\(v=office.15\).aspx#termstore](https://www.microsoft.com/library/cc262787(v=office.15).aspx#termstore), there is a maximum amount of 30,000 terms allowed per term set.

Background Jobs

The same way as SharePoint needs an infrastructure of scheduled timer jobs to perform background tasks in the system, Beezy also needs a set of processes to run in the background in order to synchronize data and execute periodic tasks.

In the server side model for on-premises installations, these background jobs are executed using standard SharePoint timer jobs. For cloud installations, these jobs run in Microsoft Azure.

There are two types of jobs: the jobs whose main task is asynchronous processing (i.e. Add-in provisioning, Beezy Backup Storage Job...) and the jobs running scheduled tasks (i.e. Beezy Weekly Email Digests Job).

Below there are the timer jobs that Beezy needs to execute:

Title	Notes	Mandatory
<i>Beezy Backup Storage Job</i>	Process scheduled by default to run every 5 minutes to feed the SharePoint activities list with the information of new activities and comments created in the Beezy database. It is also responsible of synchronizing tags from the Beezy database to the corresponding SharePoint term set. If this job is not properly running, the customer will not be able to display search results related to Beezy.	Yes
<i>Beezy User Profiles Synchronization Timer Job</i>	Process scheduled to run once a day to import to Beezy any changes in the SharePoint User Profile Service. See the User Profile Synchronization chapter for more information on this topic. Default schedule: daily 1 AM	No (only if sync needed from UPS)
<i>Beezy Domain Groups Synchronization Timer Job</i>	Process to synchronize the membership from all Active Directory groups used in Beezy to the Beezy database. Default schedule: daily 11PM	No (only if AD groups)
<i>Beezy Network Membership Synchronization Timer Job</i>	Process to synchronize Active Directory Group members to Beezy networks. Scheduled automatically after "Beezy Domain Groups Synchronization Timer Job".	No (only if AD groups)

<i>Beezy Email Broker Server Timer Job</i>	Process to handle email sending asynchronously. This behavior is optional (recommended for email servers with throttle limitations) and can be configured through Beezy settings.	No
<i>Beezy Daily Email Notifications Timer Job</i>	Process to generate daily e-mails for notification digests. Default schedule: daily 2 AM	No
<i>Beezy Weekly Email Digests Timer Job</i>	Process to generate weekly e-mails for activity digests. Default schedule: weekly Monday 3 AM	No
<i>Beezy Sites Quota Timer Job</i>	Process to collect site usages inside the Beezy site collections and send warning messages or protecting sites with quota exceeded. Default schedule: daily 5AM	No (only if site quotas enabled)
<i>Beezy Ideas Campaign Flow Timer Job</i>	Process to handle automatic events occurred in ideas campaigns. Default schedule: daily 2AM	No (only if ideas campaigns enabled)
<i>Beezy Editorial Schedule Publish Timer Job</i>	Process to handle automatic events occurred in stories and editorial content. Default schedule: every hour	No (only if stories enabled)
<i>Beezy Editorial Schedule Publish Timer Job</i>	Process to handle automatic events occurred in stories and editorial content. Default schedule: every hour	No (only if stories enabled)
<i>Beezy Audience Compilation Timer Job</i>	Process to pre-calculate audiences' membership in Beezy. Scheduled automatically after "Beezy Domain Groups Synchronization Timer Job".	No (only if stories enabled)
<i>Beezy Channel Compilation Timer Job</i>	Process to pre-calculate channels membership in Beezy. Scheduled automatically after "Beezy Domain Groups Synchronization Timer Job".	No (only if stories enabled)

Azure Architecture

The Beezy solution is installed as a SharePoint add-in connected to a remote Web site and a SQL database hosted in Microsoft Azure.

Azure Hosting

Beezy as a provider-hosted Add-in for SharePoint Online needs a Web application hosted in a cloud infrastructure.

Amongst the several options that Azure offers to host web applications, Beezy uses Azure App Services. Azure App Services is the best option to host applications for SharePoint add-ins. Admins can easily:

- scale out the application increasing the number of instances
- scale up to service tiers that offer more resources (CPU, memory, I/O, network)

They can offer auto-scaling capabilities and high availability.

The main component hosted in the Beezy App Service is the product backend, which consists of the API, the application core, the background processes and the event receivers that allow synchronization capabilities with SharePoint.

The App Service needs network connectivity with Office 365:

- From the App Service to O365, Beezy connects to the SharePoint API through Client-Side Object Model (CSOM) via HTTPS.
- From O365 to the App Service, SharePoint invokes the Receiver service endpoints available in Beezy for listening to install events, and list/library events.

Azure AD App

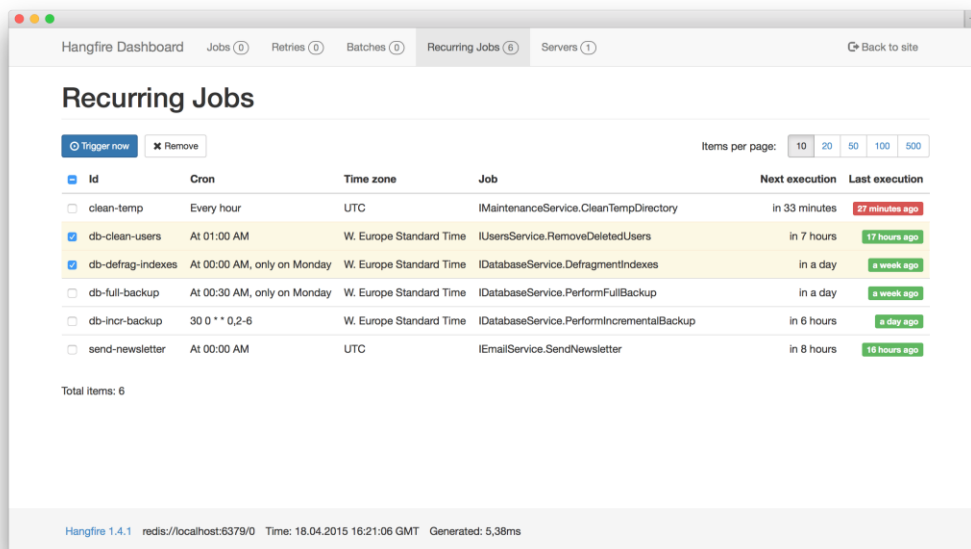
Some functionalities in Beezy require access to the Azure Active Directory to get information from AD groups. As in O365 there is no direct connection to Azure AD, Beezy uses the Microsoft Graph API to connect to it. In order to use the Microsoft Graph API it is necessary to create an Azure AD app that acts as a proxy between Microsoft Graph and Beezy.

The Beezy installation process defines the necessary steps to create and configure the app.

More information about Azure AD apps can be found [here](#).

Background Jobs

For hosting and running the background jobs, Beezy leverages [Hangfire](#), an open source third party jobs framework.



Hangfire is seamlessly integrated with Beezy, it gets deployed with the product installation package to the same Azure Web site and uses the product database to store the jobs information.

Database Server

Beezy uses Azure SQL Database as a data repository. Azure SQL is a relational database-as-a-service that delivers predictable performance, scalability, security and near-zero administration.

Azure SQL Database has several service tiers:

	BASIC TIER	STANDARD TIER	PREMIUM TIER
Uptime SLA	99.99%	99.99%	99.99%
Maximum database size	2 GB	250 GB	500 GB

Point in time Restore	Any point within 7 days	Any point within 14 days	Any point within 35 days
Disaster Recovery	Geo-restore, restore to any Azure region	Standard geo-replication, offline secondary	Active Geo-Replication, up to 4 online (readable) secondaries w/ less than 5 seconds RPO
Database Throughput Units	5	Up to 100	Up to 1000
Performance Objectives	Transaction rate per hour	Transaction rate per minute	Transaction rate per second

Beezy usually recommends Standard Tier minimum because it offers mid-level performance and built-in business continuity features including point-in-time recovery.

The performance level settings in Standard and Premium allow the customer to pay only for the capacity they need and to scale the capacity up or down as the workload changes. For example, if the database workload is busy during a specific period, the customer might increase the performance level for the database during that period and reduce it after that peak time window ends.

Monitoring

Due to the complexity of the solution and the number of users, we recommend to use a tool for monitoring the solution. There are two main tools available to use: Applications Insights and New Relic.

- [Application Insights](#), a feature of Azure Monitor, is an extensible Application Performance Management (APM) service for web developers on multiple platforms. It can monitor web applications and it will automatically detect performance anomalies. It includes powerful analytics tools to help you diagnose issues and to understand what users actually do with your app. It is designed to help continuously improve performance and usability. It integrates with your DevOps process, and has connection points to a variety of development tools. It can monitor and analyze telemetry from mobile apps by integrating with Visual Studio App Center.

- [New Relic One](#) is a development-focused platform that monitors production applications and provides deep insight into its performance and reliability. It is designed to save time when identifying and diagnosing performance issues. New Relic One tracks the load time and throughput for your web transactions, both from the server and your users' browsers. It shows how much time you spend in the database, analyzes slow queries and web requests, provides uptime monitoring and alerting, tracks application exceptions, and a whole lot more.

Front-end Architecture

The back-end of Beezy is a solution with a mixed approach between SharePoint, .NET applications and SQL Server. This has a significant impact on the front-end architecture. Below there is a description of the presentation layer for both models, on-prem and cloud.

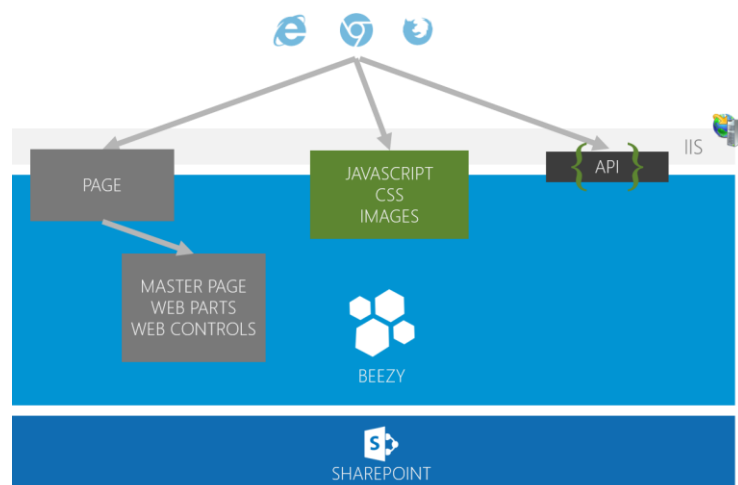
All Beezy interface components are SharePoint UI elements: master pages, pages, JavaScript files, CSS stylesheets, web parts and web controls. All these elements are stored in the Style Library of each Beezy site collection.

Most of the presentation logic takes place in the JavaScript side, where the client code is responsible to render the actual HTML templates and perform the calls to the Beezy API to retrieve and display the data.

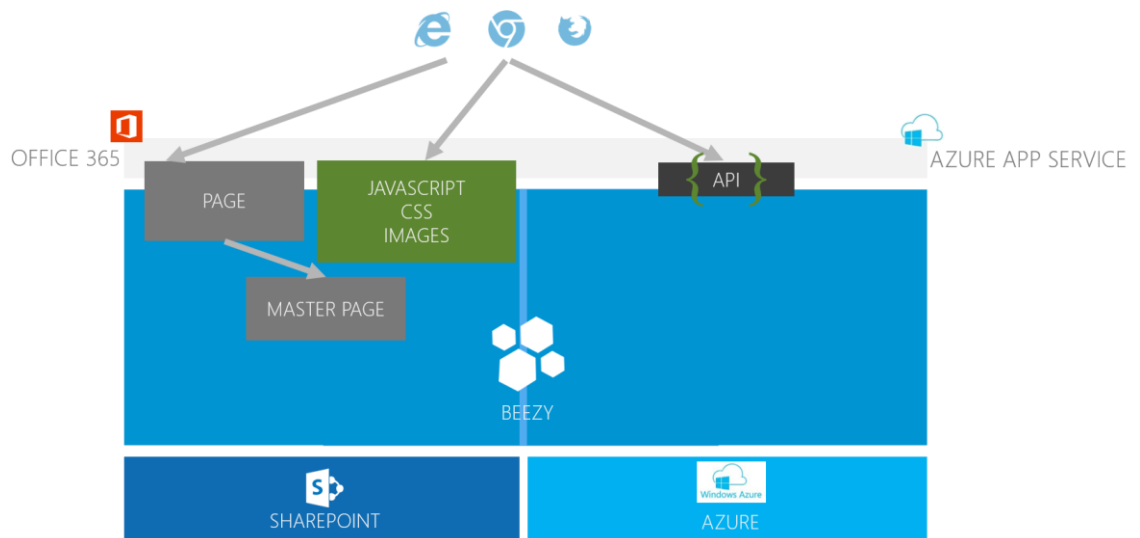
For on-premises classic installations, Web parts and Web controls execute server code inside the SharePoint context, but this server code is the minimal amount of logic needed for security trimming, redirections and context instantiation. Each web part simply renders an empty container and defines a JavaScript entry point.

For add-ins installations, every Beezy page has the UI components embedded in the page code, so every component calls the Beezy API to start querying for content and rendering the UI.

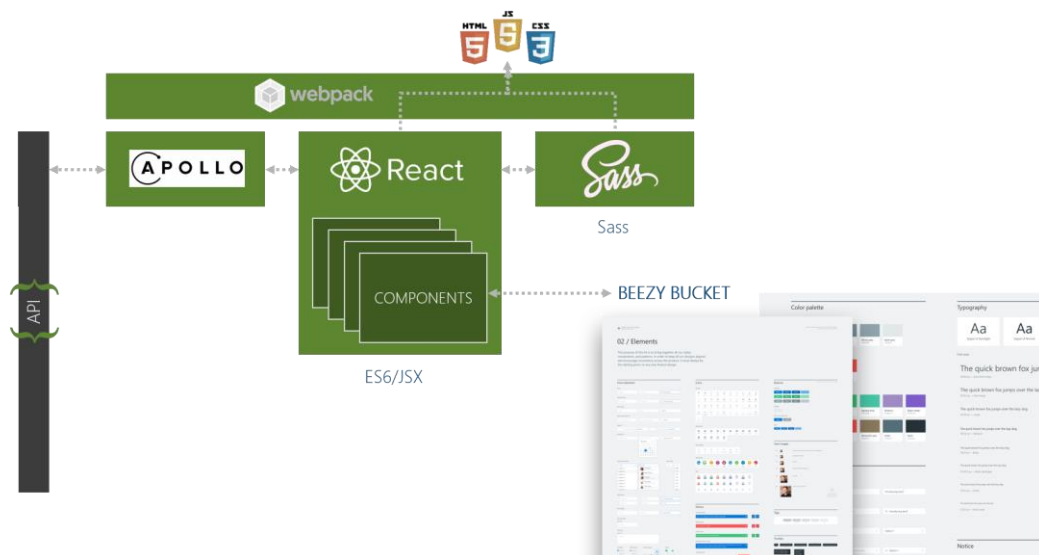
The following diagram shows the front-end architecture for on-premises:



The following diagram shows the front-end architecture for cloud:



These are the main components involved in the front-end architecture:



JavaScript Architecture

Beezy front-end side has a component-based architecture based on [React](#). It allows building encapsulated components that manage their own state, and then compose them to make complex UIs. By designing simple views for each state in the application, React can update and render just the right components when the data changes.

Since component logic is written in JavaScript instead of templates, you can easily pass rich data through the application and keep state out of the DOM. Using JSX, a syntax extension to JavaScript, you can build React elements to define the UI with all the power of JavaScript language.

Beezy uses [Apollo](#) as a platform for consuming the Beezy API. Apollo is a client framework for consuming GraphQL APIs, so components simply declare their data requirements using a GraphQL query and Apollo gets the right data to the right place, using end-to-end typing.

Beezy uses [npm](#) as a packet management system and [Webpack](#) as a module bundler. Webpack internally builds a dependency graph, which maps every module the project needs and generates one or more bundles. In Beezy, there is one bundle file for each page.

The JavaScript language specification used by Beezy is ECMAScript 6 (ES6).

Although React is the current choice for implementing the Beezy front-end components, it is worth noting that the Backbone.js framework and the MVC programming approach is still the one used in many product features. However, any new feature does not use this approach at all.

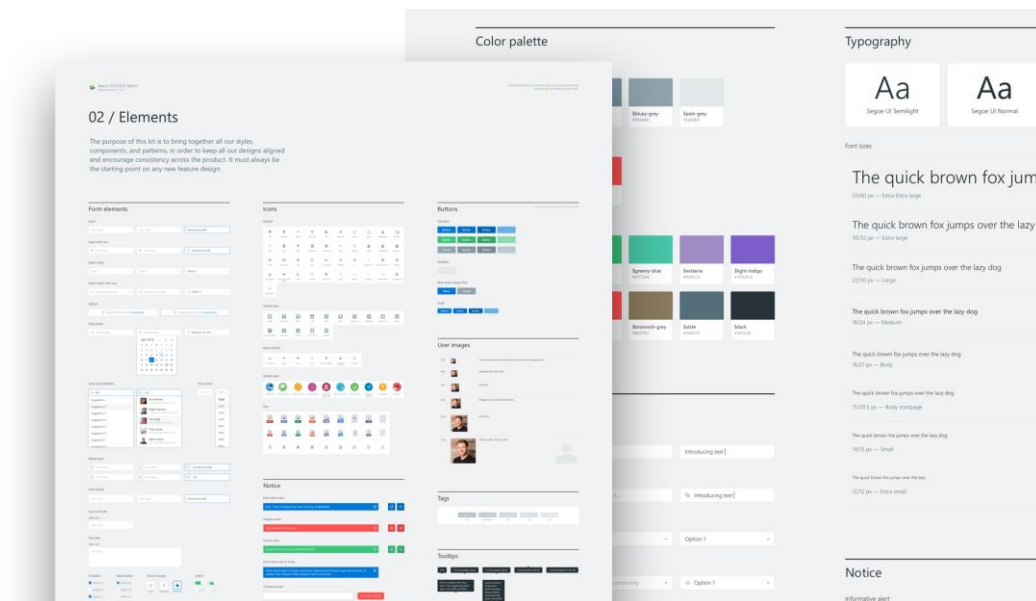
HTML/CSS

The Web rendering side of Beezy is based on HTML5/CSS3 and it is implemented using [Sass](#) preprocessor. Sass is a CSS extension that provides advanced syntax and capabilities for style sheets.

Beezy uses the [CSS Modules](#) approach to scope and isolate properly CSS rules.

Beezy has its own design system, called Beezy Bucket. A design system is a process built and maintained by a company to help develop consistent user experiences and strengthen their brand.

Every visual element of the product is a component designed with the purpose of maximum reusability, starting from the simplest ones (buttons, text fields, labels ...) and building compositions with them to produce the each application view.



Third Party Dependencies at Client

The following third party JavaScript frameworks and plugins are used at the Beezy client side.

All JavaScript third party dependencies are:

- Automatically provided by the Beezy product.
- Isolated from any other JavaScript frameworks that administrators or users might add to the SharePoint environment. The use of Webpack and bundles ensures the isolation of all Beezy JavaScript code to the rest of the code executed in each page.

Main referenced vendors (npm packages):

- *apollo*
- *babel*
- *backbone*
- *blueimp-file-upload*
- *colorthief*
- *croppie*
- *extend*
- *graphql*
- *handlebars*
- *magnific-popup*
- *masonry-layout*
- *modernizr*
- *moment*
- *node-vibrant*

- *prop-types*
- *q*
- *react*
- *react-apollo*
- *react-copy-to-clipboard*
- *react-dom*
- *react-easy-crop*
- *react-moment*
- *react-router*
- *react-truncate*
- *underscore*

Front-end Optimization

Beezy makes use of several techniques for performance optimization on the front-end side and it is important to understand them in order to be able to maintain and extend the system.

Due to the nature of the product and the team's development methodology, Beezy includes several separated files that need some pre-processing before distributing them so they do not affect the overall performance. This task reduces the number of requests to the server and the number of bytes downloaded. It also reduces the amount of work the browser must do.

During the application build (the process of creating a distribution package), all the CSS style sheets are combined into a new merged and minimized single file. The framework behind this automation is Sass.

The JavaScript files are combined into minimized files, as well. The framework responsible to build the JavaScript part is Webpack. Thus, the Beezy client can load specifically what is needed in a certain context.

In addition, to minimize the number of requests to the server all the images are grouped in sprites. The use of sprites can reduce the overall number of requests because every small image contains the colors table and has a small overhead of information. Merging several images into one sprite means reducing this overhead and sharing the colors table when done properly.

Browser Support

The following article describes the browser support for SharePoint on-premises and SharePoint online:

<https://technet.microsoft.com/en-us/library/cc263526.aspx#supportmatrix>

<https://support.office.com/en-us/article/Office-365-system-requirements-719254c0-2671-4648-9c84-c6a3d4f3be45?CorrelationId=5c473857-0319-4fbd-800f-c3e87a0bf3cf&ui=en-US&rs=en-US&ad=US>

Beezy's presentation layer supports the following browsers:

- Internet Explorer 11
- Edge
- Chrome
- Firefox
- Safari

Configuration and Settings

Beezy provides a Settings configuration page to maintain a set of core properties.

As a generic introduction, the required permissions to access the Settings page are:

- Being a site collection administrator or
- Being one of the account specified in “Administrator accounts” in Settings

These settings are stored in the Beezy database in order to have a single consistent cross-site collection storage.

There are different configuration sections, structured within the following areas:

- Basic configuration
 - License Key: section to configure the product license key number.
 - My site connection: section to configure the URL to the My Site host.
 - Security: section to configure administrator/installation accounts and global security behaviors.
 - Global features: section to enable/disable generic product features.
 - Community features: section to enable/disable features and configure settings related to communities.
 - User interface: section to configure several properties related to the product UI: start page layout, quick links, global announcements, time zone, praises, application name for e-mail templates, etc.
 - E-mail settings: section to configure properties for e-mails sent from the product (sender address, e-mail digests, etc.)
 - Gamification: section to specify how many contribution points are users awarded for the different content types.
- Advanced configuration:
 - Data classification: section to configure the informational labels to appear in communities.

- Inappropriate content settings: section to enable/disable this feature and configure its properties.
- Sites quota settings: section to enable and configure automatic site quota control. See the chapter below for more details.
- HTML snippets: section to manage HTML snippets to display within Beezy content when certain URL patterns match.
- Logging settings: section to specify properties related to tracing and logging; minimum severity logged, writing to Windows Event Log, internal audit logging, etc.
- Community settings: section to configure advanced properties for communities.
- Knowledge center settings: section to configure advanced properties for knowledge centers.
- Ideas campaign settings: section to configure administrator properties for ideas campaigns.
- Stories configuration: section to enable the stories feature in the environment.
- Pages configuration: section to enable the pages feature in the environment.
- Editorial snippets: section to manage the HTML snippets to display within the Beezy HTML editor.
- Editorial settings: section to manage user audiences for stories.
- Push notifications: section to specify the connection properties to the Azure Push Notification service for enabling mobile push notifications.
- Advanced settings: section to manage administrative configuration properties to maintenance, performance and topology.
- Application tools:
 - Profiles synchronization: section to configure the user synchronization behavior from SharePoint and which properties are synchronized.
 - Sites quota: tool for managing the quota assigned to the sites.
 - Manage inappropriate content reports: tool for managing the reports about Beezy content and the actions to take.
 - Manage sites: tool for querying and managing all the Beezy places (communities, blog, wikis, knowledge centers, etc.) in the environment.

- Manage users: tool for querying and managing all the Beezy users in the environment.
- Manage networks: tool for managing the network and site collection topology of the environment.

For getting a detailed explanation of each one of these sections, please check the Operations Guide (<https://help.beezy.net/hc/en-us/categories/200965105-Operations-Guide>).

Sites Quota

Beezy monitors every community size to report their usage using the “Sites quota” page.

The default configuration has a very high default quota (1GB) for all existing communities and new ones. The recommended approach is to monitor during a few days the actual community sizes and plan the definitive quota considering their real usage. Once you have determined the default quota, configure it through the Beezy settings page and enable the e-mail warning and read-only protection. The next time that the “Beezy Site Quotas Timer Job” runs, it will trigger the corresponding alerts and switch the read-only protection for those sites above 100%.

In the “Sites quota” page you can also customize the size individually for every community. You can customize your preferred quota and also personalize the quota warning text if needed.

Deployment/Upgrade Process

On-premises Classic Installation

Beezy is deployed by using SharePoint out-of-the-box approaches. Any SharePoint skilled professional should be able to deploy the product without Beezy's intervention.

Beezy is shipped as a set of SharePoint WSP packages and PowerShell scripts. By default the customer or the partner can execute standard command files with the needed parameters and you will get a standard installation of the product. If needed, the PowerShell scripts can be customized to enable enhanced deployment possibilities, although Beezy does not offer support for this scenario.

Deployment Components

A Beezy installation package includes the following core elements:

File name	Description
Beezy.Prerequisites.wsp	WSP solution file with third party pre-requisites for Beezy.
Beezy.wsp	Main WSP solution file with the application components.
install.ps1	PowerShell script for setting up a default Beezy installation.
uninstall.ps1	PowerShell script for removing a default Beezy installation.
upgrade.minorversion.ps1 upgrade.version.ps1	PowerShell scripts for upgrading a Beezy installation from the previous version to the current one.

Apart from these, it includes other auxiliary PowerShell scripts used by the main ones.

Deployment Process

In order to deploy a Beezy installation you need to specify a Web application and the properties of the site collection to be created (URL, site collection administrator and title).

The installation in Beezy typically performs the following steps:

- Add and deploy the Beezy.Prerequisites.wsp solution package.
- Add and deploy the Beezy.wsp solution package.

- Enable the Web application features that register the Beezy timer jobs.
- Create a site collection based on the Beezy site collection template.
- Enable the site collection scoped features that will provision all the structure and initial contents.

Beezy branding customizations for specific customers are also deployed as additional WSP solutions.

Cloud installation

For a cloud setup, Beezy is distributed as a SharePoint provider-hosted add-in. This means that there is a part of the installation in SharePoint Online and another part in Azure.

It is assumed that the customer has an Add-in Catalog site created in Office 365 in order to register all corporate SharePoint add-ins there.

Deployment Components

A Beezy installation package includes the following core elements:

File name	Description
Beezy.App.Web.zip	Web application package to be deployed to Azure.
install-online.ps1	PowerShell script for setting up a default Beezy installation.

Apart from these, it includes other auxiliary PowerShell scripts used by the main ones.

Optionally, it might contain a DLL with the branding customizations that must be deployed on the "bin" folder of the Azure website.

Deployment Process

The main steps involved in the cloud installation are the following:

- Set up a Web site in the desired Azure instance.
- Copy the Web site content to the Azure website.
- Configuration steps related to the SharePoint Online tenant (creation of the metadata term set and configuration of search properties).
- Register the Add-in in the "Apps for SharePoint" library within the associated Add-in Catalog site.
- Create a new site collection and add the Beezy Add-in to it for will provisioning all the structure and initial contents.

- Create and configure the Azure AD app that allows Beezy connecting to the Azure Active Directory.

On-premises Upgrade

The exact process of upgrading Beezy to a new version depends on the specific application features and modifications added to the application. In general, a typical upgrade process consists of the following steps:

- Upgrade the Beezy.Prerequisites and Beezy WSP solution packages (this will upgrade the application core DLL's and SharePoint artifacts).
- Upgrade the Beezy User Interface (branding) elements. In order to safely upgrade the system, Beezy provides an automatic process that removes its UI files from the SharePoint Style Library and re-provisions them.
- Upgrade the customized UI files for the specific customer, overwriting or adding those files on top of the out-of-the-box Beezy branding.
- Upgrade the database schema to prepare the Beezy database for the next version. This will usually involve the execution of a SQL script.
- Upgrade the SharePoint features and site. Beezy provides an automated upgrade script to perform the additional patches to the application in order to get the system ready. This may include activation/deactivation of SharePoint features or other required automatic operations performed to the Beezy SharePoint site.

It is important to note that upgrade processes in Beezy cannot be rolled back. This is due to the limitation of the SharePoint solution upgrading system, which does not allow standard downgrades.

Cloud Upgrade

For cloud installations, the upgrade process follows these steps:

- Override the Azure website files with the new ones from the Beezy update package.
- Browse to the Beezy upgrades page, where the admin user can start the migration process.
- The upgrade jobs run in the background and perform the necessary modifications in the database and SharePoint.

Beezy Upgrade

Current version

Product:	3.10.1.15771
Database:	3.10.0

Site collections

https://beezyenv.sharepoint.com	3.10.0.15708
https://beezyenv.sharepoint.com/sites/beezy	3.10.0.15708

Upgrade required An upgrade from version **3.10.0** to version **3.10.1** is required. Click **Upgrade** button to upgrade Beezy database and all listed Beezy site collections to the new version. Once the button is clicked, a job for each listed site collection will be scheduled. You'll be able to follow the upgrade progress from the jobs dashboard console.

Upgrade

On-premises Add-ins Installation and Upgrade

The on-premises add-ins environments follow the same installation and upgrade process than cloud environments, except that they do not need the Azure AD app.

Extensibility

Branding

Taking advantage of SharePoint capabilities, Branding is a fully customizable aspect in Beezy. Beezy allows customized branding for those customers that need a personalized look & feel for the application.

The standard branding package contains:

- Customized Beezy master pages: the customized master pages will replace the original Beezy master pages and contain the same core components but will include HTML/CSS customizations for the customer.
- Customized CSS file: this CSS file will replace the original one, which is empty, and contains all the customized classes needed for overriding or adding new styles.
- Customized JS file: this JS file will replace the original one, which is empty, and contains all the customized variables, functions or client logic needed for the UI changes.
- Customized HTML templates: the customized template files (Handlebars HBS files) will replace the original templates and contain the personalized HTML structure for the customer. The user compliance template will be commonly included in the branding package as each customer will need its own content and structure for it.
- Additional images: the image files (icons, backgrounds, etc.) needed for the customized user interface.



Classic on-premises Installations

In on-premises installations, all the needed branding files are stored in the Style Library of each Beezy site collection.

The customized branding is implemented using an additional SharePoint solution package (WSP) installed on top of the Beezy solution package. The branding package may contain web components (master pages, CSS files, JavaScript files and images) to be added to a Beezy site collection or to replace existing files. The branding package is not intended to deploy new business logic or server components, it will only provide user interface modifications.

The branding package uses a custom implementation for deploying all these files, through a feature receiver that uploads each file, overwriting it if it already exists. Just like the Beezy package, the branding package can also be versioned and upgraded.

Add-in Installations

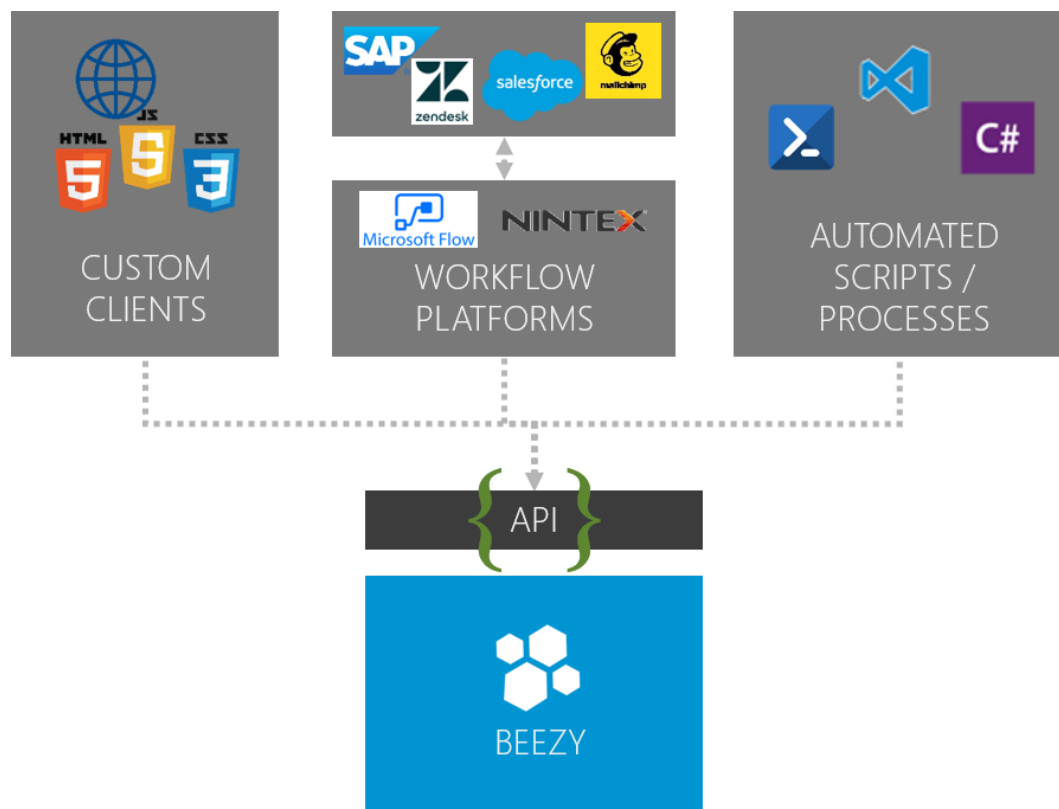
The add-in installation includes the Beezy Branding DLL which will be deployed to the “bin” folder of the Azure website for branding the Beezy user interface.

During the provisioning of the Add-in, the branding files are copied to the Style Library of each Beezy site collection.

API Extensibility

As Beezy offers the 100% of its functionality through an interoperable interface, the Beezy API enables seamless integration scenarios with third party components, tools or platforms. As a product, Beezy offers almost unlimited integration possibilities with existing platforms in the market.

The range of integration scenarios is very wide, from automating community creation to create Beezy activities linked to a specific external content (i.e. a new SAP object, a file modified in an external document storage, a certain step occurred in a Nintex workflow). One of the interesting possibilities of the Beezy activity model is the ability to extend the activity types and metadata that they can store.



Multi-language Support

The Beezy User Interface is available in multiple languages. However, the application does not currently manage content language or translations in all its features, just a subset of them (stories, pages, apps, megamenu).

A large part of the user interface is handled by the SharePoint platform itself through the SharePoint language packs (in on-premises environments).

According to the SharePoint standard behavior, the language displayed to the user is the one defined in the user's browser. By default, a SharePoint site that has defined alternate languages will be shown in the language defined in the browser's language settings. A user can override this behavior by editing their SharePoint user (Language and Region).

About the Beezy product, the translated literals are managed in two different files: a JavaScript file (named `beezy.i18n.js`) for all pure UI literals and a server resource file (`Beezy.xx_xx.resx`) for literals used during provisioning and installation.

For all three different installation models, Beezy delivers the language translations embedded in the product itself service package and the UI translations (`beezy.i18n.js`) are copied to all SharePoint site collections during the provisioning process. The server resource file (`Beezy.xx_xx.resx`) is hosted in the Azure/IIS side (for add-ins) or the SharePoint hive folder (for classic on-premises).

It is possible to override the UI literals for any language in branding packages, creating `beezy.branding.i18n.xx-xx.js` files.

Governance

In this section there are some generic aspects and indications that will apply when defining a governance plan with a customer.

Logging and Monitoring

On-premises Installations

Beezy logs to the two different standard sources allowed by SharePoint on premise:

- Unified Logging System (ULS): all events
- Windows Event Log: only errors

You can disable the error logging to Windows Event Log using the Beezy Settings page. The error events are logged to Windows Event Log with the standard SharePoint severity levels:

- Error
- Critical

Also, you can configure the minimum severity level traced to the ULS by using either the event throttling configuration in Central Administration (Diagnostic Logging) or the Beezy Settings page. The events are traced with the standard SharePoint severity levels:

- Verbose
- Medium
- High
- Monitorable
- Unexpected

You can enable/disable deep audit logging for API calls and internal method calls using the Beezy Settings page. This is only recommended for temporary auditing or troubleshooting purposes.

Beezy relies on standard SharePoint monitoring tools, as SharePoint is the underlying platform of the Beezy stack. As Beezy logs detailed audit information to SharePoint, it is possible to collect usage and performed operations using ULS parsing/processing tools.

Cloud Installations

Instead of relying on the SharePoint logging infrastructure for on premise (ULS), SharePoint Add-ins need to trace and log using different approaches.

The Beezy web application hosted by Azure traces the events to an internal registry, as traces do not need to be sent to SharePoint (only errors will be logged through the SharePoint APIs). This internal registry uses Log4Net as a standard framework to configure where and how the log events are traced.

Please also refer to the [Azure Monitoring](#) chapter to see more details regarding frameworks for monitoring errors and exceptions.

SharePoint Governance

As described in Microsoft official documentation, Beezy already defines its own strategy about the main governance topics related to IT (<https://technet.microsoft.com/en-us/library/cc262883.aspx>):

- Security, infrastructure, and web application policies
 - Maintenance of the system and infrastructure: according to how Beezy implements the deployment and integration of SharePoint site collections and sites, it can be managed with standard platform procedures and added to the existing IT policies. As Beezy works on top of SharePoint, the application can leverage all its features, including auditing, document retention, document discovery, web application policies, etc.
 - Maximum upload file size: document storage in Beezy relies in standard SharePoint document libraries, which are subject to the file size limits managed by centralized IT policies
 - Control of fine-grained permissions: Beezy as an application makes minimal use of broken permission inheritance, only when it is mandatory. This implies avoiding the use of item-level permissions and recommending all customers and implementations to apply the same criteria.
- Data protection (backup and recovery): the next chapter describes the standard procedures for backup and restore of Beezy.
- Site policies: according to how Beezy implements the deployment and integration of SharePoint site collections and sites, and the way that Beezy supports a multiple site collection scenario, any customer

can enable site policies in communities and collections of communities, i.e. closing and deleting sites automatically or by running a workflow.

Other features like self-service site creation could be integrated with Beezy, but with some additional customization effort, leveraging Add-in Model implementation approaches like

http://blogs.msdn.com/b/richard_dizeregas_blog/archive/2013/04/04/self-service-site-provisioning-using-apps-for-sharepoint-2013.aspx

- Quotas: SharePoint quota configuration and templates can be used at site collection level. When it comes to communities (subsites), Beezy offers its own solution of quota management. Please check the chapter "Configuration and settings" > "Sites quota" to see more details.

About monitoring, Beezy logs detailed audit information related to internal method calls, API calls, exceptions, warnings and tracing information from all processes to SharePoint in order to collect usage and performed operations. In addition, the application includes alert and notifications for certain features like community approval and site quotas.

For other application management and customization policies, please check the chapters [Application Lifecycle Management](#) and [Deployment/Upgrade Process](#).

Backup and Recovery

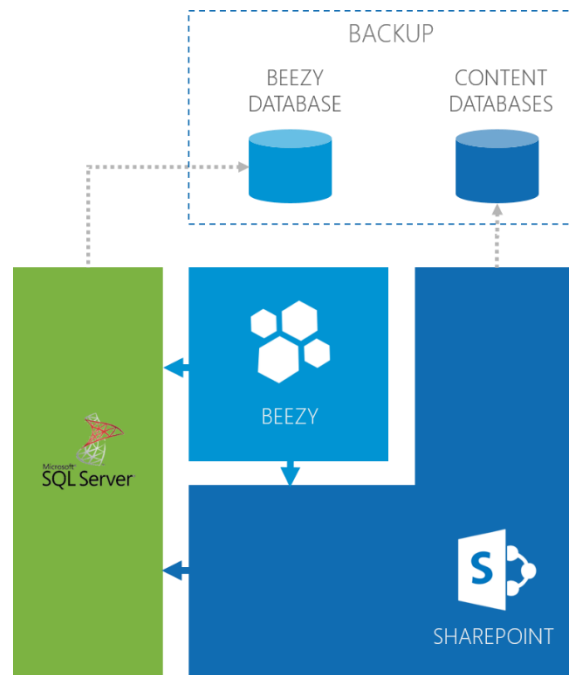
On-premises Backup Procedure

The main objective of the backup for the Beezy application is to be able to recover a snapshot of the whole environment in an isolated recovery environment. This environment is the basis for performing all disaster recovery operations.

In order to have a complete snapshot of Beezy at a certain moment of time, these elements need to be in place:

- A backup of the SharePoint content databases for all site collections where Beezy is deployed (using standard SharePoint backup procedures)
- A backup of the Beezy database (using standard SQL Server backup)

Depending on how the global SharePoint backup is performed, the content databases will be part of the complete backup process of the SharePoint farm.



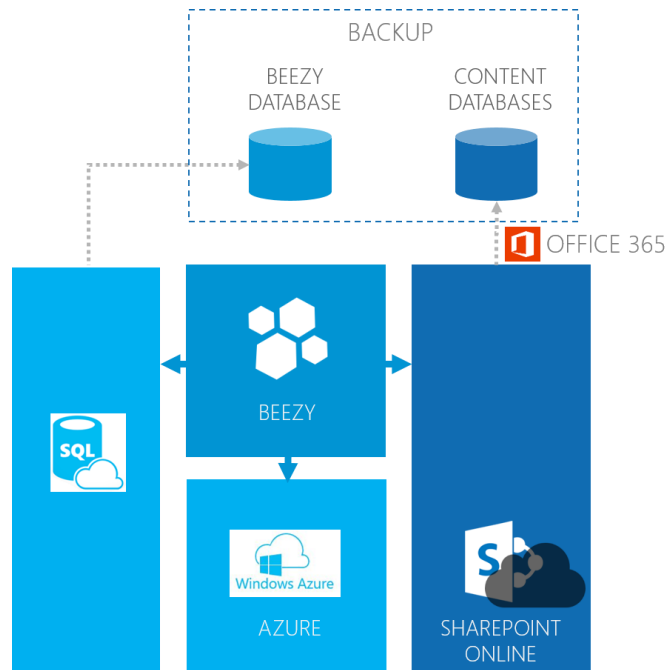
Cloud Backup Procedure

The main objective of the backup for cloud is the same as on premise, but the procedures in Office 365 support for backup and restore impact on the overall process.

In order to have a complete snapshot of Beezy at a certain moment of time, these elements need to be in place:

- A backup of the SharePoint content databases for all site collections where Beezy is deployed. Office 365 performs these backups automatically in the background.
- A backup of the Beezy database (using standard Azure SQL backup)

It is important to remark that, according to the Office 365 architecture, the site collection backups cannot be recovered outside the O365 instance where they were backed up.



It is also worthwhile mentioning that there are third party solutions available to manage backup and restore processes in a more convenient way than the offered by the default Office 365 service, for example AvePoint's DocAve Online.

On-premises Full Recovery

If the entire solution needs to be recovered from a catastrophic failure, these are the steps to perform a full recovery for on premise installations.

- Set up a new SharePoint environment. You can skip this step if you need to recover only the site collections, not the whole SharePoint farm.
- Execute the Beezy installation as specified in the Beezy installation guide. You can also skip this step if you need to recover only the site collections, not the whole SharePoint farm.
- Restore the SharePoint site collections from the backed up SharePoint content databases (using standard SharePoint restore procedures).
- Restore the Beezy SQL database from the backup (using standard SQL Server restore)

Cloud Recovery Strategy

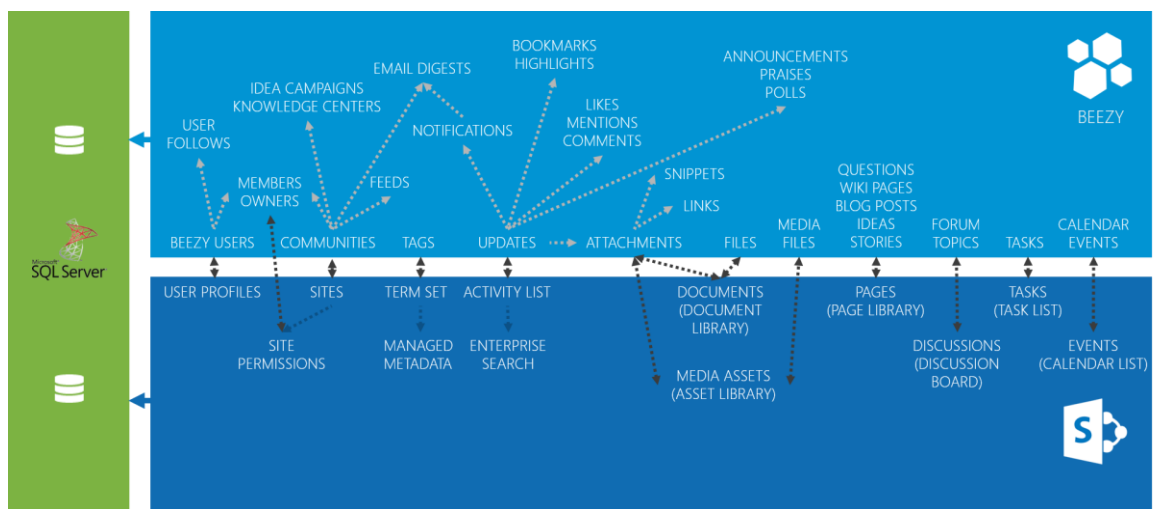
If the entire solution needs to be recovered from a catastrophic failure, these are the steps to perform a full recovery for cloud installations.

- Set up a new Azure environment (you can reuse an existing Azure instance).
- Restore the SharePoint site collections through the standard recovery process for Office 365.
It is important to remark that, according to the Office 365 policies, restoring a site collection backup requires opening a support ticket to the Office 365 team. Site collection backups can only be recovered in the same instance, and the restore procedure can take two or more days.
- Restore the Beezy SQL database from the backup (using standard Azure SQL restore procedure).

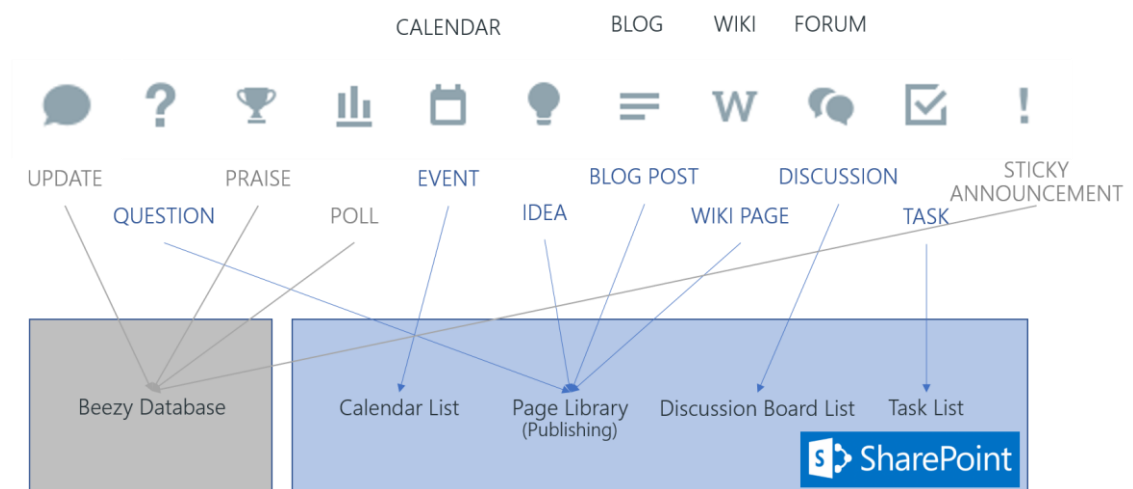
Other Topics

Data Architecture Diagrams

The following diagram provides a deeper view of detailed data objects from the collaboration and social model, where they are stored and how are mapped between the Beezy and SharePoint context.



The following diagram indicates for every type of activity in the Beezy sharebox what the primary storage is:



This table describes the types of Beezy activities not created with the sharebox:

	Start page newsfeed	Place newsfeed	My profile newsfeed	
FILE	✓	✓	✓	Documents uploaded via Files tool
EXTENDED IDEA	✓	✓	✓	Ideas created in ideas campaigns
ADAPTIVE/ACTION CARD	✓	✓	✓	Automatically created via workflows
PAGE		✓		Created in corporate modules
STORY		✓		Created in editorial modules
ITEM	✓	✓	✓	Created via SharePoint lists and libraries
SHARED ACTIVITY	✓	✓	✓	Created via sharing other activities
SHARED APP	✓	✓	✓	Created via sharing apps

Third Party Components for Server

Apart from SharePoint and SQL Server, Beezy makes use of some other components to provide the whole functionality to its users. Below you can find the list of components used by Beezy with some notes attached to them.

Component	Notes
<i>NHibernate</i>	<p>NHibernate is an object-relational mapping (ORM) solution for the Microsoft .NET platform. It provides a framework for mapping an object-oriented domain model to a traditional relational database. Its purpose is to relieve the developer from a significant portion of relational data persistence-related programming tasks. NHibernate is free as open-source software that is distributed under the GNU Lesser General Public License. NHibernate is a port of the popular Java O/R mapper Hibernate to .NET.</p> <p>Current version installed: 3.2.0.4000</p> <p>http://nhforge.org/</p>
<i>NHibernate SysCache</i>	<p>NHibernate Second Level cache provider. This type of cache is persistent, lives across multiple requests and is used by all sessions concurrently. Roughly speaking, when an object instance is fetched from the database all</p>

	<p>values of the object are stored in the cache. When the same object is requested again, NHibernate will dehydrate the object using the values found in the cache which are associated to that particular identifier of the object.</p> <p>Current version installed: 3.1.0.4000</p> <p>http://nuget.org/packages/NHibernate.Caches.SysCache2</p>
<i>Fluent NHibernate</i>	<p>Fluent NHibernate offers an alternative to NHibernate's standard XML mapping files. Rather than writing XML documents (.hbm.xml files), Fluent NHibernate lets you write mappings in strongly typed C# code. This allows for easy refactoring, improved readability and more concise code.</p> <p>Current version installed: 1.3.0.717</p> <p>http://www.fluentnhibernate.org/</p>
<i>AutoMapper</i>	<p>Object-object mapper library, which works by transforming an input object of one type into an output object of a different type, based in a set of conventions to make mapping configuration and usage easier.</p> <p>Current version installed: 6.0.2</p> <p>https://github.com/AutoMapper/AutoMapper</p>
<i>CryptoLicensing</i>	<p>.NET solution to add licensing, copy-protection and activation capabilities to .NET software.</p> <p>Current version installed: 6.0.0.0</p> <p>http://www.ssware.com/cryptolicensing/cryptolicensing_net.htm</p>
<i>Unity (Microsoft Patterns and Practices)</i>	<p>Lightweight extensible dependency injection container with support for constructor, property, and method call injection.</p> <p>Current version installed: 4.0.1</p> <p>http://unity.codeplex.com/</p>
<i>WCF REST Starter Kit</i>	<p>Microsoft suite of helper classes and extension methods to build REST interfaces based in WCF protocols.</p>

	<p>Current version installed: 1.0.0.0</p> <p>http://msdn.microsoft.com/en-us/library/ee391967.aspx</p>
<i>Log4Net</i>	<p>Tool to help the programmers output log statements to a variety of output targets.</p> <p>Current version installed: 1.2.15.0</p> <p>https://logging.apache.org/log4net/</p>
<i>Newtonsoft.Json</i>	<p>.NET library to serialize and deserialize any .NET object from/to JSON format.</p> <p>Current version installed: 9.0.1</p> <p>http://www.newtonsoft.com/json</p>
<i>HtmlAgilityPack</i>	<p>.NET library to parse, navigate and process HTML data efficiently and easily.</p> <p>Current version installed: 1.4.9.5</p> <p>http://html-agility-pack.net</p>
<i>Handlebars.Net</i>	<p>.NET extension to support Handlebars (Mustache) templating format in .NET applications.</p> <p>Current version installed: 1.7.1</p> <p>https://github.com/rexm/Handlebars.Net</p>
<i>Hangfire</i> (add-ins only)	<p>Framework to perform background processing in .NET and applications without additional processes or services.</p> <p>Current version installed: 1.6.12</p> <p>https://www.hangfire.io/</p>

In classic on-premises installations, all these third party assemblies are included into the Beezy.Prerequisites WSP solution file. Therefore, if you uninstall Beezy from your environment, you should keep this WSP deployed in the farm until you are sure that none of the components are needed by other applications. The Beezy uninstallation scripts will remove only the Beezy WSP package, but not the Beezy.Prerequisites WSP package.

In add-in installations, the third party assemblies are deployed into the Azure/IIS infrastructure, as additional dependencies of the cloud service and worker roles. In this case, it will not necessary any removal policies, as there will not be any conflicts with other applications.

Licenses and maintenance are fully covered by Beezy, meaning that if any of the third party components fail Beezy will release a new package with the fix when the third party component solves the problem. Beezy handles the relationship with the third parties.

Third party components for the client side (JavaScript) are described in the chapter "Front-end architecture".

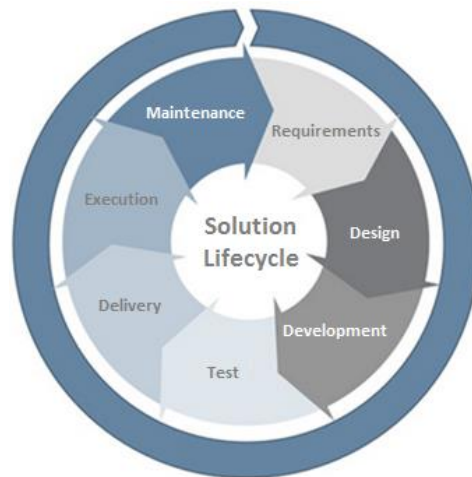
External APIs

Beezy needs to consume from two external APIs:

- **Embedly:** Embedly is a set of front-end tools that provide rich content extraction from URLs. Beezy needs Embed.ly to extract snippet previews from external URLs and rendering them as part of the newsfeed, in each post.
The Embedly API is called from the JavaScript components at Beezy, using HTTPS requests.
Embedly is a paid platform, and customers pay for the use of Embedly's processing engine. There are three API modes in Embedly; Beezy uses the Extract mode, which allows extracting from URLs the article text, topics, and retrieving more meta-data about articles, blog posts, and stories.
There are two options to support this external API:
 - Beezy acting as a middle service provider, maintaining an Embedly instance for the customer.
 - The customer acquiring their own Embedly instance and configuring it in the Beezy settings.
- **Bing Maps:** The Microsoft map engine offers a set of available APIs for different platforms. Beezy needs the geocoding feature (finding coordinates for a certain address) for the functionality of displaying the employee location in the My Site header. In order to find and display the coordinates, Beezy makes use of the Bing Maps REST Services, through the license mode Bing Maps for Enterprise Light Known User.
There are two options to support this external API:
 - Beezy acting as a middle service provider, maintaining a Bing Maps license for the customer.
 - The customer acquiring their own Bing Maps license key and configuring it in the Beezy settings.

Software Development Lifecycle

Beezy takes care of all phases of the Software Development Lifecycle (SDL) by using methodologies and tools for the different phases involved.



- Requirements and planning: Beezy uses a set of tools for managing requirements and handling properly the change management: Atlassian Jira and Office 365.
- Design: Beezy uses different standard design methodologies like ULM prototyping, relational model design with Visual Studio, Code-First design, and implementation of mock-ups and visual prototypes with Invision, Axure, or directly on HTML or SharePoint. In addition, for the definition of use stories, acceptance criteria and test cases, Beezy uses Jira and TestRail.
- Development: Beezy has different templates and methodologies in order to configure optimized development environments, using standard tools and frameworks for programming (Visual Studio, Visual Studio Code), productivity (ReSharper, Stylecop, SPCop), source control and versioning (BitBucket, Git), build and automation (Jenkins, Webpack, MSBuild).
- Test: Beezy uses unit testing for both server and client development, and takes advantage of standard server frameworks like NUnit, NCover, Moq and BDD (Behavior Driven Development) test frameworks like Specflow and Selenium.

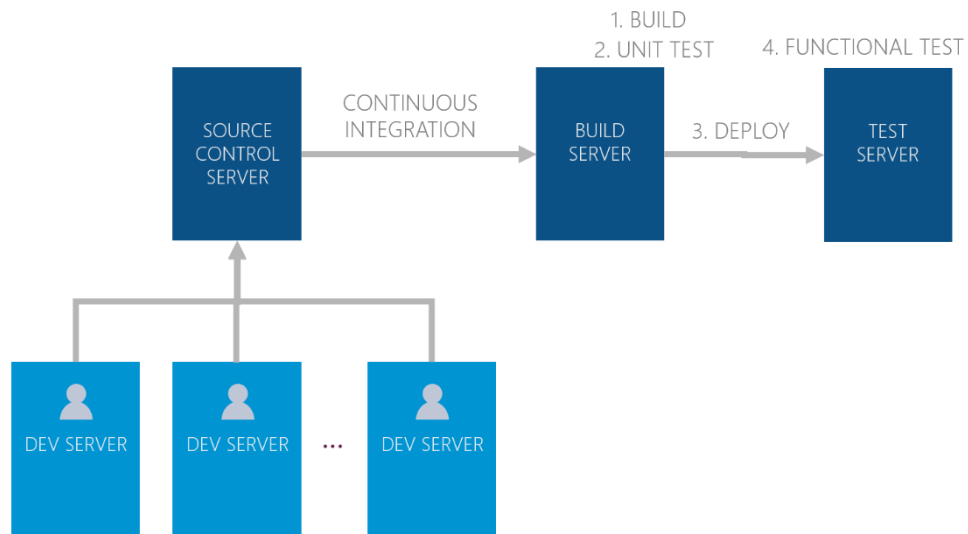
Regarding the client side, the JavaScript code not directly related to user interface is covered by unit testing combining frameworks like Jest and react-testing-library.

In addition, Beezy performs Load & Performance testing for each release (using Visual Studio Load Tests). The aim of these tests is to maintain a consistent user response time below 3 seconds in a system under high stress.

- **Delivery:** Beezy provides the deliverables using standard methodologies for publishing and release like SharePoint WSP packages, SharePoint add-ins and automation of installation processes with PowerShell scripts.
- **Execution and maintenance:** Beezy uses different tools for issue tracking / reporting and change management like Zendesk, Atlassian Jira, SharePoint and the Beezy product itself as a collaboration platform, allowing fluid interaction with the customer and monitoring cases and tasks.

The development of Beezy is based on a continuous integration (CI) approach. Using a distributed source control system the Beezy team is able to develop different features at the same time and integrate them into a main development branch, through pull requests that imply a code review by all team members. The main development branch is continuously integrated, this means, built, tested and deployed in each new code revision (version of source code), to ensure its stability.

The continuous integration occurs every time the source control receives a new revision. The project code is built and the unit tests and coverage are checked. After that, the binaries are deployed to a test server and the functional test is performed against it.



SharePoint Limits and Boundaries

It is important to take some of the well-known SharePoint limits into account along with the expected usage of the platform by the end users.

There is information below regarding each limit with relation to the current architecture, the desired architecture and alternative solutions.

Recommended amount of sub sites per site (2,000)

The interface for enumerating subsites of a given Web site does not perform well as the number of subsites surpasses 2,000. Similarly, the All Site Content page and the Tree View Control performance will decrease significantly, as the number grows.

Impact of this limit

The customer can go over this limitation by distributing the communities (subsites) in different site collections.

Recommended amount of items per list returned by a query (5,000)

Specifies the maximum number of list or library items that a database operation, such as a query, can process at the same time outside the daily time window set by the administrator during which queries are unrestricted.

Impact of this limit

The customer should keep the number of SharePoint items in a list or library under this number. This applies to blog posts, wiki pages, tasks, files, questions, etc.

Recommended amount of SharePoint groups per site collection (10,000)

Above 10,000 SharePoint groups, the time to execute operations is increased significantly. This is especially true of adding a user to an existing group, creating a new group, and rendering group views.

Impact of this limit

Beezy needs 2 SharePoint groups for each community and moderated community. The total number of groups needed by Beezy will be the sum of:

*2 * number of communities*

The customer can go over this limitation by distributing the places (subsites) in different site collections.

Recommended amount of groups a user can belong to (5,000)

This is not a hard limit but it is consistent with Active Directory guidelines.

Impact of this limit

This limitation affects the total number of SharePoint groups a user can belong to, including:

- Being a member of a community*
- Being an owner of a community*

Anyway, the customer can go over this limitation by distributing the places (subsites) in different site collections.

Recommended amount of users in a SharePoint group (5,000)

This is not a hard limit, but impacts on performance in operations like fetching users to validate permissions. Having up to 5,000 users (or Active Directory groups or users) in a SharePoint group provides acceptable performance.

Impact of this limit

This limitation affects the total number of individual members that a community can have. This is a significant limit only for public communities, where users can join freely. The customer will need to monitor public communities with high demand or usage to avoid exceeding this limit.

Recommended number of metadata elements (30,000)

You can have up to 1,000 term sets in a term store, up to 30,000 terms in a term set and, in total, up to 1,000,000 items in a term store.

Impact of this limit

In order to have the tags available for SharePoint search, Beezy will synchronize the tags from the Beezy repository to the SharePoint term set in an asynchronous process.

Do not exceed the SharePoint recommended storage limit (200GB for on premise / 1TB for online) per site collection

The recommended content database limit for SharePoint is 200GB for on premise and 1TB for online. Check details to overcome this limitation in <https://technet.microsoft.com/en-us/library/cc262787.aspx#ContentDB>

Impact of this limit

The customer can always go over this limitation by distributing the communities (subsites) in different site collections, stored in different content databases.