ZaaS

Zip as a Service

Note: One month of programming saves 1 day of architecture

Elevator Pitch

The idea is to transfer VERY FAST between (one location and multiple others) multiple files from a folder.

Description

Zip As A Service (ZaaS) is a cloud-based application designed to provide compression and decompression services over the internet. It allows users to upload files or directories, which the service then compresses into a zip file. Conversely, users can also provide zip files for the service to decompress, extracting the contents for download or direct manipulation online. ZaaS aims to offer a convenient, efficient, and scalable solution for handling zip operations without the need for local resources or software, catering to both individual and enterprise-level needs. It could include features like API access for automation, secure file handling, and support for various compression formats beyond zip, such as rar, 7z, etc.

Also , it can zip a whole bunch of files that are stored in different locations on PC and can be downloaded as a single zip file.
The files could come from different locations on the PC and the user can select the files from different locations and then the service will zip them and provide a download link to the user.

Also automation can be seen as a feature where the user can schedule the zipping of files at a particular time and the service will zip the files and provide a download link to the user.

Core Features

- File Compression: ZaaS allows users to compress files and directories into zip archives.
- File Decompression: ZaaS can decompress zip files, extracting the contents for download or manipulation.

- Cloud-Based: The service operates over the internet, eliminating the need for local software or resources. However, it can also be integrated with local systems.
- Scalability: ZaaS is designed to handle varying workloads, from individual users to enterprise-level demands.
- API Access: The service may offer an API for programmatic access, enabling automation and integration with other systems.
- Secure Handling: ZaaS ensures secure file handling, protecting user data and maintaining privacy.
- Support for Multiple Formats: Zip by default. At enterprise request , ZaaS may support other compression formats like rar, 7z, etc.
- Batch Compression: ZaaS can compress multiple files from different locations on the PC and provide a single zip file for download.
- Automation: Users can schedule file compression tasks to run at specific times, with the service providing download links upon completion.
- User-Friendly Interface: ZaaS offers an intuitive interface for easy file upload, compression, and download operations.
- Customization: Users may have options to customize compression settings, file naming, and other parameters.

Users

Number of users is initially low . Estimating that users that download the same file, at peak, will be like between 100- 1000 / country .

Compliance ?? TBD

Security ?? TBD

- File integrity
- Least privilege principle

Data in transit / Data at rest ?? TBD

SLO / Quantity

Quantity will be negotiated with each enterprise. However, we estimate no more than 100 GB at first per day .
For each 100 MB will be a max time to upload of 10 minutes (depending also on the enterprise internet connection). After the transfer is complete, any 100 MB will be compressed in 1 minute.
Expiry time : usually 1 month after upload (depending on subscription - premium, normal, free)

What is this cap of 100 GB / day necessary:

For startup = we need to assess how big the infrastructure will be (compute power, network , …)
Equality of users : the cap will be subscription based (premium , normal, free)
Affects scalab***ility***

From client perspective: a max cap to be seen ,even for the free subscription.

Additional Features

- Encryption: ZaaS could offer encryption options for securing compressed files.
- Deliver files to client - decompress and streaming by browser (Content type stream)
- File Management: Users may have tools for organizing, renaming, or deleting files within the service.
- Versioning: ZaaS could support versioning of compressed files, allowing users to track changes over time.
- Support for Multiple Formats: In addition to zip, ZaaS may support other compression formats like rar, 7z, etc
- Collaboration: Users might be able to share compressed files with others, enabling collaboration on projects.
- Integration: ZaaS could integrate with cloud storage services, email clients, or other applications for seamless file handling.
- Monitoring & Reporting: The service may provide insights into compression/decompression activities, usage statistics, and other metrics.
- Custom Workflows: Users could create custom workflows for specific compression tasks, combining multiple operations into a single action.
- Notifications: ZaaS might offer notifications for task completion, errors, or other events related to file compression.
- Compression Algorithms: Users may have options to choose different compression algorithms or levels based on their needs.
- File Preview: ZaaS could provide a preview of compressed files before download, ensuring the correct content is included.
- CAP per day : number of files or total size for upload .will make user to shift towards plan with larger caps. (X GB/day limit non functional requirement)

- One feature that could be required for enterprise to adopt this service is the ability to bring their own encryption keys. data at rest usually it's encrypted (functional requirement)
- agent to sync on-prem with cloud
- Looking for the name of the files directly on cloud
- Sync with later modification with files
- Prevent man in the middle attack - have a checksum validation
- Networking: VPN site to site

Performance considerations

- splitting project (or files) greater than a given threshold allow to distribute the compression workloads to multiple machines (spot or on demand) being time and cost effective (non functional requirement)
- CDN to cache archives this will ensure faster consecutive downloads
- Major cloud providers offers storage with HA but if the service is global we need to take in consideration latency (region to region) From architecture point of view this may translate in a non functional requirement to have replication between buckets and multi region access point to let the service to choose the best route
-

—-

—----------

TODO:

ADR - de identificare
https://github.com/joelparkerhenderson/architecture-decision-record
https://github.com/joelparkerhenderson/architecture-decision-record/tree/main/locales/en/examples/4-day-work-week

Citit C4 Model: https://c4model.com/

Vazut

- ArchiMate Modelling pentru Enterprise Architecture
- Togaf Standard
- PlantUML

Identificat componente - de gindit cui se adreseaza diagramele respective

https://github.com/horeabiris/ZaaS

Nu va uitati aici
Diagram Viorel :
https://excalidraw.com/#json=loQfiURRWJJxfeFH5AUNN,D-DLDW9kfdoRV61rSge8cQ

Interactiuni componente

https://mermaid.live/edit

Andrei
—-

https://mermaid.live/edit#pako:eNqFU01vwjAM_StRTiDBH-hhB0BIkzZpA7RJoxxC44JFG3dpAkKI_76E0FJoJ3roh_2e_fycnnhCEnjEhU62aCAxVsNwDUbEirlro8kWTEK5M1TMQcleCXoPur_8GLMDmi2bYgYlWxAbAXMAs4pVoHogJsBySqlmvZNCQ3pKmQS9Yqiate95JaRY83z60qnNCazqGQSLAr-1G6eXZGRlfzn2DzaDBHAPQXJLp272u0JvLauKbc5k1JNY7pxGN5jYPOWkiKru48GvKqUneN9DGLEWJdw4k1EHq2Nt11mamzPE9ggHR3etH43YY0YNK74c8I0SkbX8vxb-dwUzEPJ-A152bX-z569tdPy0oI8XVNMaX-1R6IPMO-tv-CCs6U-4-5MZzdhw-MLeIn_cqvIp1mHdHfZLCWH_Fi0u4VEUdhUSnlknwkGpHAoO10k_fGfCD1gJ5gOeg84FSve3ni4DcbOFHGIeuVcp9C7msTo7nLCG5keV8MhoCwPu9rHZ8igVWem-bOGOEkxQbLTI62gh1A9R9X3-A2DIYhQ

```
architecture-beta
    group desktopSend(server)[PC with Files To Be Sent]

    service mofo(server)[MonitorFolder] in desktopSend
    service sefi(server)[SendFiles] in desktopSend
```

group apiWrite(cloud)[Cloud Receive Files]

service refi(server)[ReceiveFiles] in apiWrite
service refiDB(disk)[StorageFiles] in apiWrite
service fiin(server)[FileInfo] in apiWrite
service fiinDB(database)[FileInfoDB] in apiWrite

group desktopReceive(server)[PC to view info]

service vilofi(server)[ViewLocalFiles] in desktopReceive

group apiRead(cloud)[Cloud Info Files]
service qufi(server)[QueryFilesInfo] in apiRead
service vifi(server)[ViewFiles] in apiRead

mofo:R --> L:sefi
sefi:R --> L:refi
sefi:R --> L:fiin
fiin:T --> B:fiinDB
refi:T --> B:refiDB

vilofi:T --> B:qufi
vilofi:T --> B:vifi

Container

What container do you think is necessary ?

https://c4model.com/abstractions/container

https://static.structurizr.com/workspace/36141/diagrams/Containers.png

Constraints:

TimeToMarket
Budget
Performance
Itty (Scalability , maintainability, observability…)

Service / Daemon on client (? component /scheduling / monitor zip files (create, update, delete))

Scheduling Service / Daemon on server (perform cleanup , payments , etc)

Compression / Decompression

Encryption / Key rotation container

WebSite GUI

(

this could be components ?

listing the zip files / crud /

/ user / company management / payments

)

Cost Calculator / Adviser / Container (? do not have feature, please add)

Storage

(this could be components ?

Storage for zip files

Storage for data (user, company, token)

Storage for logs / open telemetry /

)

API Gateway (definition of API and access)

Notification Service (? part of API Gateway ?)

Payment Service (? part of API Gateway ?)

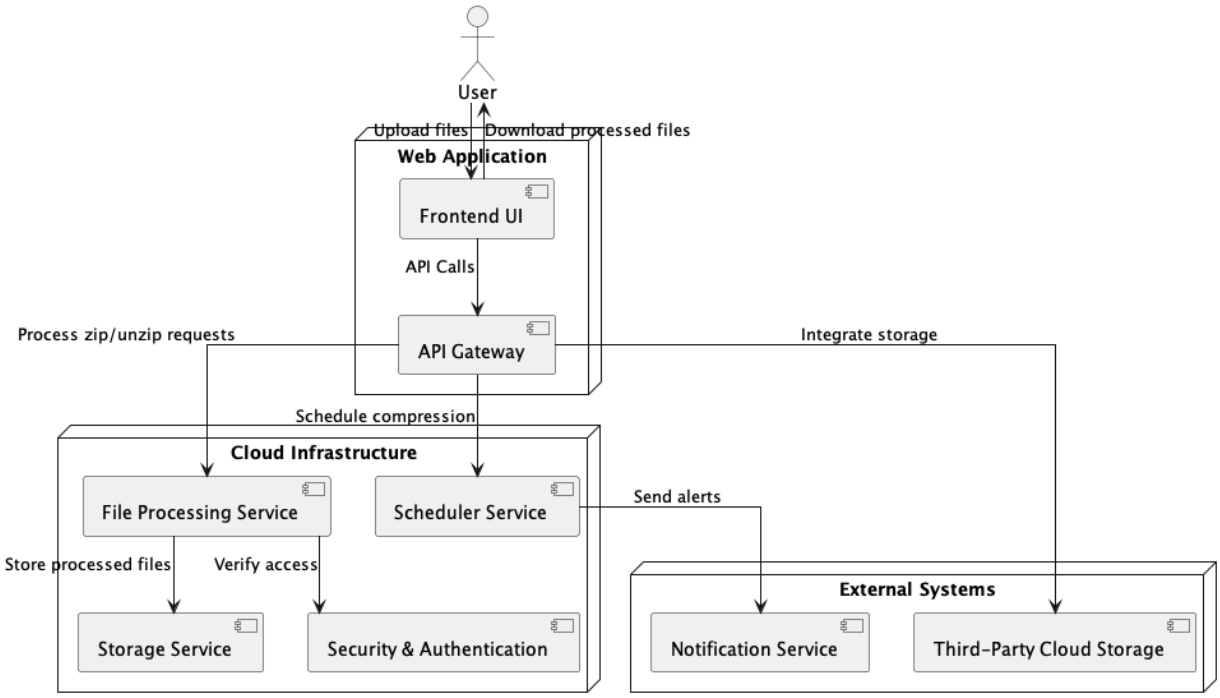
Desktop Application (for display status)

Monitoring / Observability / Tracing(?)

SIEM (maybe too advanced , maybe cu Elastic)

Maybe later
Container differences

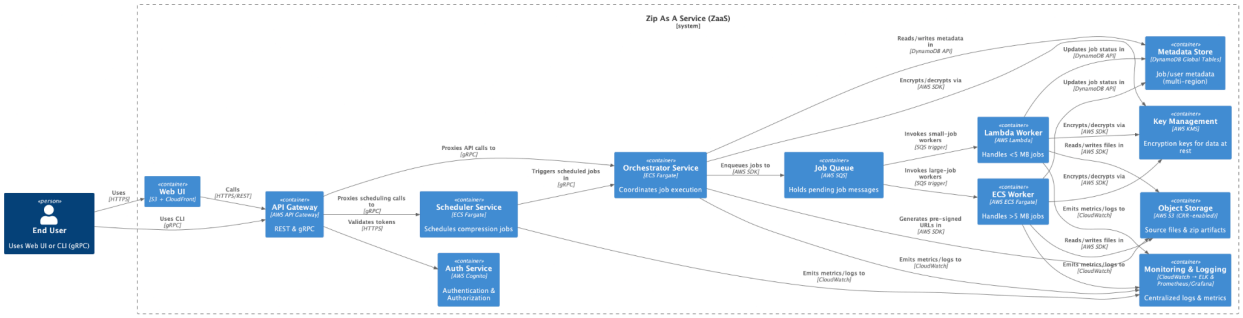
CHAT GPT - Software architect agent



\

Andrei
https://mermaid.live/edit#pako:eNqNlF1v2jAUhv-K5SuQoCLhOxeTNlgrtK5DpdOkES4O9mmwSOzIdkpTxH-fw1foqKC-iXPs98k577GzpkxxpAEFzRbCIrOZxvocLYSSuBFplaWEo1IalU5Q8opB_YK6Oh3EAqUlSpN7xSAm4wFZCbsgtyJGQ54U-YbECewslDtUMQqxYOieHFh8ZE324SFgoiTZoWdEyNMvn1MWnFe4MMvq9AHtSullmc2tijnq6wiOkB7TGO52EkjTWDCwQslLhKKE4JHU61_IfeByKddOvTNWaYhwX2KFxSrjruJdlOzD5mOXmMHSI7ZAnsVCRgfRNrf3-HNG0d6yZypJNRrjCiNDZOXbp1Aol6jVEfZdMp2nhUnkB-bkUdnSsWsoYyscLMzB4NGLTwkTpeZWH3P4qaRwisKUX_MiCHMRC5uTJw3MRWcXkAfHd216ESaDWLyhrghpUUu01emomAHblvVRg1YYZeJk_x-cT9wlIne_R9tqSui5FlIRwYn263hE7sDiCvKrWqlODsaDsuJ5f1zfHY1LhBROCGPIk-IuXxP_PwslrdEEdQKCu1_IuoiG1C4wwZAGbspBL0Mayo3bB5lVk1wyGlidYY1mqTsBOBQQaUgOwRQkDdb0lQZtz7vx_X631et7fq_fb9ZoToNmo33j-R3P6_X8bqvT6mxq9E0pp2_c9Lrthht-p9HtNvy2v6X93S7u4K7P0YIGzxAb9xbpIun9irvZqAcqk5YGXnPzDyZ3tjw

```
architecture-beta
    group desktopSend(server)[Client or Local PC with Files To Be Sent]

        service sedacl(server)[Service Daemon Client] in desktopSend
        service hdd(disk)[Network or Local Folder] in desktopSend
        service deap(server)[Desktop application] in desktopSend
        sedacl:R --> L:hdd

    group storageService(cloud)[Storage Services]

        service scse(server)[Scheduling Service] in storageService
        service code(server)[Compression Decompression] in storageService
        service enkero(server)[Encryption Key Rotation] in storageService
        service st(database)[Storage] in storageService
        service moobtr(server)[Monitoring Observability Tracing]in storageService


    group visualizer(internet)[Interaction]

        service wegui(internet)[WebSite GUI] in visualizer
        service apiga(internet)[API Gateway] in visualizer
        service nose(server)[Notification Service] in visualizer
        service pase(server)[Payment Service] in visualizer
```


Zip As A Service (ZaaS)
[system]
«person»
End User
Uses Web UI or CLI (gRPC)
Uses
[HTTPS]
Uses CLI
[gRPC]
«container»
Web UI
[ST + CloudFront]
Calls
[HTTPS/REST]
«container»
API Gateway
[AWS API Gateway]
REST & gRPC
Proxies API calls to
[gRPC]
Proxies scheduling calls
to
[gRPC]
Validates tokens
[HTTPS]
«container»
Auth Service
[AWS Cognito]
Authentication &
Authorization
«container»
Scheduler Service
[ECS Fargate]
Schedules compression jobs
Triggers scheduled jobs
to
[gRPC]
«container»
Orchestrator Service
[ECS Fargate]
Coordinates job execution
Reads/writes metadata
in
[DynamoDB API]
Encrypts/decrypts via
[AWS SDK]
Enqueues jobs to
[AWS SDK]
«container»
Job Queue
[AWS SQS]
Holds pending job messages
Invokes small-job
workers
[SQS trigger]
Invokes large-job
workers
[SQS trigger]
Generates pre-signed
URLs in
[AWS SDK]
Emits metrics/logs to
[CloudWatch]
Emits metrics/logs to
[CloudWatch]
«container»
Lambda Worker
[AWS Lambda]
Handles <5 MB jobs
«container»
ECS Worker
[AWS ECS Fargate]
Handles >5 MB jobs
Updates job status in
[DynamoDB API]
Updates job status in
[DynamoDB API]
Encrypts/decrypts via
[AWS SDK]
Reads/writes files in
[AWS SDK]
Encrypts/decrypts via
[AWS SDK]
Emits metrics/logs to
[CloudWatch]
Reads/writes files in
[AWS SDK]
Emits metrics/logs to
[CloudWatch]
«container»
Metadata Store
[DynamoDB Global Tables]
Job/user metadata
(multi-region)
«container»
Key Management
[AWS KMS]
Encryption keys for data at
rest
«container»
Object Storage
[AWS S3 (CRR-enabled)]
Source files & zip artifacts
«container»
Monitoring & Logging
[CloudWatch + ELK &
Prometheus/Grafana]
Centralized logs & metrics

Components

Actor (user or system)

- system

UseCase

Actor (user or system)