New issue

# [WatsonX] GenAI Flow - Online mode #36122

Open
marius-danciu opened this issue on Mar 15 · 71 comments

## Comments

Member

### marius-danciu commented on Mar 15 • edited by ginaulak

**Tracker:** #39313

Currently wx support py-functions deployment that are designed for scoring tasks. These functions are used for online scoring usecase and they have certain traits:

- They are synchronous - meaning that they can return a single response.
- They use an opinionated protocol for request and response tailored more for tabular data often use for ML cases.

With the addition of foundation models the number of usecases explode into various directions as now we are talking about text LLM models, image, sound, video etc. Some of these use-cases require streaming, some don't.

With this git item we are introducing a new type of py-function. Details can be found here: https://ibm.box.com/s/5jlxc72jtwvc01xvphanm8675dumrlp4

The traits of this new function type are:

- No opinionated request and response payload. Users can use whatever payload they want.
- The function can choose to stream down the response or not.

---

Initial tasks #36122 (comment) ; for detail breakup and see tracker #39313

marius-danciu assigned dejan, marius-danciu, svagaral, vegoutha, otucker and mromila on Mar 15

marius-danciu added the watsonx.ai Related to Watsonx.ailabel on Mar 15

Member

## dejan commented on Mar 15

In addition, I would propose to deliver 'payload' as a dictionary that has:

1. 'input' - the object that is taken from REST API that represents the portion of JSON that contains question, past history etc.
2. 'token' - the IAM or JWT token that was sent to the REST API via the 'Authorization' header. This is important so that inferencing inside the py_function can use credentials set using the token, not the API key. This will improve performance and inferencing will be done in the name of the API caller, not the API key creator.

MemberAuthor

## marius-danciu commented on Mar 15 • edited

@dejan I really don't think we should propose any payload whatsoever. This is the main point - **to be un-opinionated**..
The payload may not even be JSON. Could be binary, XML whatever the author wants. On the other hand, the functions that our tooling will generate yes in that case the implementation will assume a certain structure for input and output because it needs to work end to end.

Member

## dejan commented on Mar 15

Well, I was talking about the function itself. We need the input (which could as you say be JSON turned into python dictionary, or something else), and token received through the headers. I understand that by passing headers in you are already giving me that indirectly, so maybe we don't need token (I can just fish out the token from headers, 'Authorization' header to be precise).

MemberAuthor

### marius-danciu commented on Mar 15

Correct. The token would be retrievable from the headers object. The 'input' is going to be extracted from the `payload` object.

MemberAuthor

### marius-danciu commented on Mar 23 • edited

@dejan I was thinking on what we discussed yesterday and I don't think that a function should have a streaming and non streaming endpoint. It is one function and one endpoint and this is easier to reason with.

Once the stream functions are used, the HTTP response will be in SSE mode and the function return value will be ignored.

- The function implementation can switch between stream and non stream mode based on its own protocol or headers. We are not opinionated here either.
- The function caller based on the function documentation would reason between stream and non stream mode.
- Once the function returns the connection will be closed
- We are not opinionated on how the last message in the stream looks like. This is up to the function implementation. This is why the stream object does not need a function like `close()` or `emitEos()`

# The main action items + rough estimates:

- Support a new asset type called genai_flow - **5 PD**
- Implement new CRUD endpoints `/v1/genai_flows` - **14 PD**
- Support the ability for the caller to document the request and response shape (possibly with examples). This should be json. - **5 PD**
- Deployment APIs allow the ability to query for deployments of `genai_flows` - **2 PD**
- Deployment service will support SSE via the Stream object. If the stream was not used, the function return value will be used as response. This can be a tuple of the body and response headers. If there are not headers provided in the response `application/json` content-type will be assumed. - **10 PD**
- Feature Add for CSF - **? PD**

ginaulak self-assigned this on Mar 25

marius-danciu changed the title [WatsonX] GenAI Functions on Apr 2

marius-danciu mentioned this issue on May 14
Gen AI Flow Deployments #36670
Open

marius-danciu changed the title [WatsonX] GenAI Flow - Online mode on May 14

Lukasz-Cmielowski added the WML-client-python label on May 17

Lukasz-Cmielowski assigned Dorota-Laczak on May 17

Member

## ginaulak commented on May 23 • edited

WML Deployment Team , Minutes of Meeting today: participants @ginaulak @marius-danciu @julianpayne @vegoutha @sribhaum and others

- Showcased a small POC demo on online and SSE
- SSE : need to check further how flask handles SSE and if we need to explicitly set `eos_event`
- We have not found a way to implement same `interact()` for both online and SSE based on return type. This is mainly due to flask + gevent requirement that a generator is returned for SSE. We could have 2 function for this as an alternative generate & generate_stream (function name may change)
- For online and SSE ; if user has configured Task Credentials, we will expose the token filename in env var USER_TOKEN_FILE_PATH
- USER_TOKEN_FILE_PATH=//token.txt . This token is generated from Task Credentials and is refreshed at regular interval by sidecar. File will not exists if Task Credentials is not configured.
- For batch jobs; Task Credentials will be a pre-requisite
- There is a case made for `init() / init(token)` which will be additionally called during asset Deploy . This is technically not needed , but could be an "optional" feature which could be called if callable init is present. It can provide a net way of code organisation for user. The token here in init(token) is token generated from Task Credentials as an alternative to the env var USER_TOKEN_FILE_PATH . Small additional effort to implement. Not concluded if it should be implemented, to discuss further.
- We need two more deployment endpoints e.g. "/ml/v4/deployments/:deployment_id/generate" and "/ml/v4/deployments/:deployment_id/generate_stream" . (endpoints name TBD) . To discuss further

- In generate/generate_stream/interact (payload, headers, stream) : we will inject `headers` with `bearer token` that was used at the time of calling "/ml/v4/deployments/:deployment_id/generate" and "/ml/v4/deployments/:deployment_id/generate_stream"
- Two options were discussed to implement batch code. (1) call it via python module `genai_module.batch_job(payload)` (2). To call it in subprocess . `subprocess.call("python", "mygenapp.py")`
    - Option (1) preferred as error handling is better here. In genai_module.batch_job(payload) , payload is the deployment jobs payload format at job deployment time.
- To check with python client team if ibm-watson-machine-learning already interfaced with Milvus and other vector databases. To follow up with @Lukasz-Cmielowski
- For batch job, the deployment service will not download / upload data_references from deployment_jobs payload, unlike existing ML/DL frameworks
- The genai_flow asset content can be either compressed (like python function) or uncompressed (like python script). We choose to not compress , but does not really have strong opinion here. It will be a single file for genai_flow asset. If there are multiple files required as zipped / compressed , we need to explore this in future .. either as genai_asset itself or.. implement with package_extensions (workaround). Is this zipped python installable? if so which is the main file etc.. are some initial thoughts
- asset_type to be managed by WML repository. The metadata for genai_flow is not concluded yet. To discuss further
- To conclude what are the additional packages required to be part of the runtime image . langchain, llamaindex , more? . To follow up with @abmanish
- To check with Watson Studio team , and agree on which Runtime to use RT23.1 is latest on Cloud, RT24.1 is latest on CPD . This task is for Cloud so most likely RT23.1 ; To check with WS if these additional LLM packages can be offered in the RT of our choice. To follow up with @ALEXLANG
- That a new software spec may be an overkill. We can use existing one runtime-23.1-py3.10 (if using RT23.1)
- Custom Library will be supported along with this. But known issue of package installed before server startup issue will not be handled along with this item.
- To come up with sizing including Batch. ETA 24May
- bonus: The deployment service will not be running multi worker for this; There is also need to implement infrachange to switch from `threads` to `gevent` async to make stream work with flask.

```
pass
```

Member

## dejan commented on May 23

For online and SSE ; if user has configured Task Credentials, we will expose the token filename in env var

A lot of online/SSE responses can be generated using the access token passed to the REST API of the deployment. We should not always assume that the flow function runs using an API key for several reasons:

1. Creating a token from the API key adds overhead
2. API key is potentially for a different identity than the one used for REST API token provider. There are use cases where using the caller's token is desirable, even necessary.

Member

## dejan commented on May 23

To check with python client team if ibm-watson-machine-learning already interfaced with Milvus and other vector databases.

Why does this matter? Gen AI flows should have no preference as to what runs in them - Milvus and other vector databases should be no concern for it. I don't understand why this is even discussed.

Member

## dejan commented on May 23

```
subprocess.run("pip install langchain", shell=True)
print(f"langchain installation done.")
```

I have been told that there is a better way of creating a custom configuration that defines packages needed by the python function than running subprocess like this.

Member

## dejan commented on May 23

Side comment: due to the upcoming release of watsonx.ai Flow engine (WatZen), the term 'flow' may be taken. While using it in the UIs is easier to change, references in asset types,

function/variable names may be more sticky and we should be careful as we may need to change it in a breaking way.

@abmanish

MemberAuthor

**marius-danciu commented on May 24**

A lot of online/SSE responses can be generated using the access token passed to the REST API of the deployment. We should not always assume that the flow function runs using an API key for several reasons:
Creating a token from the API key adds overhead
API key is potentially for a different identity than the one used for REST API token provider.
There are use cases where using the caller's token is desirable, even necessary.

For online cases I agree that the user's token passed in the inference request is needed. Task-credentials should not be mandatory for online scenarios but it is required for batch. creating the token from task_credentials won't happen at each request as this would be cached and refreshed only when it is about to expire. So this won't be generated on each request.

Why does this matter? Gen AI flows should have no preference as to what runs in them - Milvus and other vector databases should be no concern for it. I don't understand why this is even discussed.
Agreed again. The py client does not care about the milvus interaction. The genai function can write to a Milvus or other vector DB. The WML python client does not have a mandate to provide APIs for such DBs.

I have been told that there is a better way of creating a custom configuration that defines packages needed by the python function than running subprocess like this.
We already have a mechanism for installing custom python packages. We don't expect this to change at least not now. I do agree with the proposed option 1 that the batch implementation should be done in a specific py function that we call rather than a "free" script.

On the streaming aspect using python's yield operator means that we no longer need the stream object.

ginaulak mentioned this issue on May 24
RT24.1 - genai additional packages - langchain llama-index #38099
Open

Member

## otucker commented on May 27

How coupled is this to a specific runtime? If a user creates these genAI flows and we decide to deprecate a runtime, are they going to have to redo/migrate these again? I'd like to see if we can have something that's not going to add to the inertia of any given runtime (which typically only last 12-18 months).

Member

## ginaulak commented on May 28 • edited

Updating slack discussion i had with @marius-danciu

- Create WML Deployment currently has `custom: object` in POST body https://cloud.ibm.com/apidocs/machine-learning#deployments-create . We will expose this in init

init is optional, the service will check if a callable by that name is available in the module first. Or we could just make it requirement for genai_flow?

- SSE user code only handles the `data` part of the stream event. The service takes care of the `id` and `event`. After the data from user code is exhausted, the service ends the stream with last `event: eos` before closing the connection.
  e.g.
  User stream generator

Member

## ginaulak commented on May 28

Hi @otucker

How coupled is this to a specific runtime? If a user creates these genAI flows and we decide to deprecate a runtime, are they going to have to redo/migrate these again? I'd like to see if we

can have something that's not going to add to the inertia of any given runtime (which typically only last 12-18 months).
@Lukasz-Cmielowski mentioned that langchain and llama-index(not approved yet) will be part of python SDK starting RT23.1 and above . The level of how much this genai_flow deployment is coupled to the runtime would be similar to python-functions. For the most part it would require them to update their genai_flow asset software specification on runtime discontinuation . This is still subject to compatibility of the package versions between the runtimes , for whichever packages genai_flow asset is consuming (user code)

Member

## Lukasz-Cmielowski commented on May 28

@otucker as @ginaulak mentioned this is coupled to RT. RT 23.1 is targeted at this stage (tracked here: #38099 (comment)).

The open question is RT 24.1 release on cloud - if that would happen soon I would skip RT 23.1. Keep us honest here.

marius-danciu mentioned this issue on May 29
watsonx.ai onboarding request for Mistral-Large NGP-TWC/watsonx-ai-onboarding#91
Open
0 of 22 tasks complete

MemberAuthor

## marius-danciu commented on May 29

As genai_flows will be assets in CAMS we need to design the CRUD APIs that WML Repository will implement. But before that we need to define the metadata properties for geai_flow. Here is an initial draft ptoposal

@abmanish can you think of other meta-data infomration that would be needed here ?
@julianpayne @ginaulak . any thoughts ?

ginaulak mentioned this issue on May 29
[WatsonX] genai_flow contents and python SDK interface #38174
Open

Member

## ginaulak commented on May 29 • edited

**WML deployment centric tasks**

**Open Items:**

- - decide on the deployment endpoint for online e.g. `https://us-south.ml.cloud.ibm.com/ml/v4/deployments/:deployment_id/genai`
- - decide on the deployment endpoint for SSE .e.g `https://us-south.ml.cloud.ibm.com/ml/v4/deployments/:deployment_id/genai_stream`
- - decide on naming of callable functions `init/generate/generate_stream/generate_batch` #38174
- - to conclude on asset name `genai_flow/function` #36122 (comment)
- - Will be on RT24.1  RT24.1 WML will have overhead #35575 we will need to do this first (most of the task can be done in parallel)

**Notes:**

- Runtime 23.1 / RT 24.1 whichever is going to be use will include langchain & llama-index (subject to approval) via python SDK #38099 (comment)
- We will use the same set of default T-shirt sizes used by python function in phase 1 . Hardware_spec from the asset will be implemented in phase 3 . The hardware_spec can be specified at deployment time currently.
- CUH billing basis will be similar to python function

**Dependencies:**

- WML Repository /ml/v1/genai_flow proposal here #36122 (comment)

**Phase 1: Tasks Online support**

- Deployment Manager: Support new runtime `genai_flow` as single tenant, Support Online
- Deployment Manager: (Misc) Handle activity tracker, Consumption (CUH), metrics, segments, hibernation, Patching, RMQ-cams-Events, Scaling and RU
- Deployment Manager: Deployment activatation after hibernation (Score Handler) - New dep endpoints /func /func_streaming
- Deployment Manager: New Repo endpoints with utils-libs integration. (lib..) /ml/v1/genai_flow/
- Sidecar: support `custom` component of deployment payload #39191
- Sidecar: changes to cascade Header information for online and stream.
- Sidecar: Payload Logging

- Sidecar: New dep endpoints + new repo endpoints (utils-lib-go)
- Runtime: get base image ready with updated python SDK (python SDK is expected to come bundled with langchain and llama-index)
- Runtime: infrastructure - move to gunicorn flask gevent
- Runtime: new runtime to support genai_flow online - new endpoints
- Runtime: header in arguments
- configuration changes to nginx etc. to support the endpoints. - new endpoints
- Documentation updates
- sample notebooks for documentation
- FVT

**Phase 2: Tasks Batch and SSE support**

- Deployment Manager: Support Batch data_reference. Inline is not supported
- Sidecar : support SSE
- Runtime : support SSE
- Runtime : support batch data_reference
- FVT

- No Plan to support Hardware_spec from asset metadata

Member

## dejan commented on May 29

#36122 (comment)

Do not get attached to the `genai_flow` name or the term `Gen AI Flow`.

👍1ginaulak reacted with thumbs up emoji

Member

## ginaulak commented on Jun 6

@dejan @abmanish @marius-danciu
Please keep us posted, as and when the terminologies to use are confirmed. Thanks

Member

## Lukasz-Cmielowski commented on Jun 6

Can we just stick to `Gen AI Functions` or sth like that. Remove the confusing part - Flow ?. Looping in PM team -> @acasaletto.

MemberAuthor

## marius-danciu commented on Jun 6

I thought we agreed with @abmanish and @dejan that the name is genai_flow. We need a stable terminology to be able to actually implement things. We just cannot afford to have endless conversations on this, not if we want to deliver things.

P.S.
@Lukasz-Cmielowski my initial proposal was genai_function but then (if I recall correctly) @abmanish mentioned that the agreement is genai_flow. I'm also a bit skeptical about "flow" since there is nothing that actually imposes a descriptive flow. Sure one can also argue that a function just implements a flow :)

👍1Lukasz-Cmielowski reacted with thumbs up emoji

Member

## dejan commented on Jun 6 • edited

I agree that there is nothing to suggest flows in how functions are implemented - you may as well return “hello world” from it and it would be valid.

I am fine with “Gen AI function” for the term and “genai_function” for the asset type.

WatZen is strongly against anything else in the product being called “flow”.

👍1marius-danciu reacted with thumbs up emoji

Member

## acasaletto commented on Jun 6

Let me confirm naming - will come back ASAP.

In terms of the WatZen team being strongly against anything else in the product being called "flow" - that ship has well and truly sailed years ago. We have Modeler Flows and Orchestration Pipelines both of which are creating Flows in a section called Flows...

In terms of the reason this initially was proposed as "Gen AI Flows" is that this is the Deployment mechanism for anything that we create - from simple functions to fully fledged flows - whether they are created using open source or Watzen

👍1ginaulak reacted with thumbs up emoji

Member

### dejan commented on Jun 6

Please do - I don't feel strongly about naming other than we should pick one and stick to it.

Collaborator

### abmanish commented on Jun 9

I agree that PM and Marketing across watsonx and WatZen need to get to an agreement on the naming.

My preference will be to stick to GenAI Flow. A GenAI Flow is a generic term that can represent a Flow/logic implemented using LangChain, LlamaIndex, Watsonx flows or anything else. I was debating if we should just call it watsonx_flows but then realised that our product names keep changing and it is better to have something not tied to a product name.

Member

### dejan commented on Jun 9

Absolutely, keep product names out of resource names if you want them to last :-).

Member

### ginaulak commented on Jun 12

wrt to the genai_function / genai_flow content and how it will be saved to repository with python SDK as discussed here #38174 ; This is how it will be

```
def deployable_function(params=None):
    # Below can be imported while deployment creating
    from ibm_watsonx_ai.foundation_models.utils.decorators.genai_function import
init,generate,generate_stream,generate_batch

    @init
    def myinit(custom_object):
        ...

    @generate
    def mygenerate(payload, headers):
        ...

    @generate_stream
    def mygenerate_stream(payload, headers):
        ...

    @generate_batch
    def mygenerate_batch(scoring_payload):
        ...

    # The ordering of returned tuple, does not matter for us; we will use the
<function>.__name__ to figure out
    return myinit, mygenerate, mygenerate_stream, mygenerate_batch
```

Saving to repository:

```
client.repository.store_genai_function(function=deployable_function, metadata)
```

The decorator names and the expected inner function **name** (renamed by the decorator) will be:

- init
- generate (not generate_text)
- generate_stream
- generate_batch

Member

## ginaulak commented on Jun 12

wrt to the deployment endpoints; we will stick with:

```
https://us-south.ml.cloud.ibm.com/ml/v4/deployments/:deployment_id/func
https://us-south.ml.cloud.ibm.com/ml/v4/deployments/:deployment_id/func_stream
```

Other options discussed:

`func` and `func_stream` preferred over this.

```
https://us-south.ml.cloud.ibm.com/ml/v4/deployments/:deployment_id/genai
```

`https://us-south.ml.cloud.ibm.com/ml/v4/deployments/:deployment_id/genai_stream`
This below one could be confused with BYOM

```
https://us-south.ml.cloud.ibm.com/ml/v4/deployments/:deployment_id/generate
https://us-
south.ml.cloud.ibm.com/ml/v4/deployments/:deployment_id/generate_stream
```

Member

## julianpayne commented 29 days ago

@marius-danciu can we use `flow` for the API paths as `func` is really very non-descriptive?
`https://us-south.ml.cloud.ibm.com/ml/v4/deployments/:deployment_id/flow`
`https://us-south.ml.cloud.ibm.com/ml/v4/deployments/:deployment_id/flow_stream`

MemberAuthor

## marius-danciu commented 24 days ago

To me,`flow` is way less descriptive than `func` or `genai` because there is nothing really that describes an actual flow. Also the asset type would most likely be `genai_function` which would seem disconnected from `flow` endpoint.

Member

## ginaulak commented 24 days ago

Updating slack discussion between @ginaulak @julianpayne @marius-danciu @vegoutha

1. Endpoint agreed upon will be

- `https://us-south.ml.cloud.ibm.com/ml/v4/deployments/:deployment_id/genai` : will serve generate
- `https://us-south.ml.cloud.ibm.com/ml/v4/deployments/:deployment_id/genai_stream` : will serve generate_stream

2. On successful deployment of `genai_function` in WML ,

- we will return urls
  for `generate` and `generate_stream` under `.entity.status.inference` as showned below
- `.entity.deployed_asset_type` will indicate `genai_function`

```
{
  "entity": {
    ...
    "deployed_asset_type": "genai_function",
    "name": "my genapp test",
    ...
    "status": {
      "inference": [
        {
          "url": "https://wml-
fvt.ml.test.cloud.ibm.com/ml/v4/deployments/fc831263-8685-4337-a1ff-
0249280329fe/genai"
        },
        {
          "url": "https://wml-
fvt.ml.test.cloud.ibm.com/ml/v4/deployments/fc831263-8685-4337-a1ff-
0249280329fe/genai_stream"
          "sse": true
        }
      ],
      "state": "ready"
    }
  },
  "metadata": {
    ...
  }
}
```

julianpayne added the WML-Swagger label 18 days ago

Member

## julianpayne commented 18 days ago

@ginaulak we will need these new APIs documented in the swagger so please add the API definition here (or a linkt= to where I can see a detailed description of the API)?

Comment

**Pipeline**

No Workspace yet - Create One

Assignees

dejan

marius-danciu

svagaral

vegoutha

ginaulak

otucker

mromila

Dorota-Laczak

Labels

deployment-squad1watsonx.aiRelated to Watsonx.aiWML-client-pythonWML-deploymentv4 GAWML-Swagger

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

Notifications

Customize

Subscribe

You’re not receiving notifications from this thread.

13 participants