

The Crucial Role of AI Evaluations

Confoo.ca 2026



About Michael Dawson

Senior Principal Software Engineer

AI & Data Science on the Ecosystem Engineering team at Red Hat

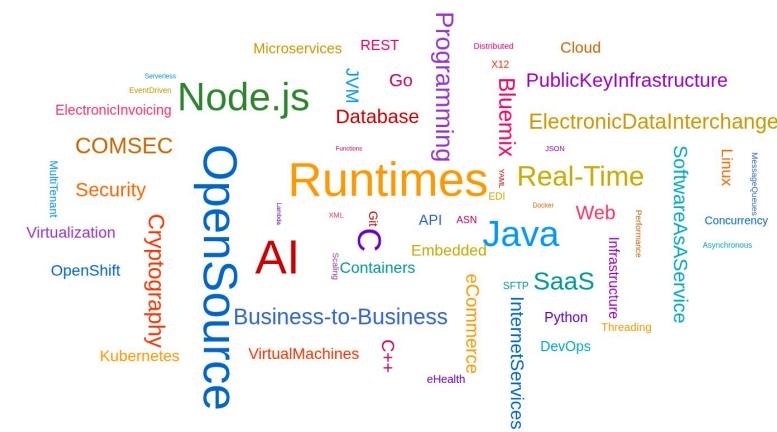


BlueSky: @mhdawson.bsky.social

Twitter: @mhdawson1

GitHub: @mhdawson

Linkedin: <https://www.linkedin.com/in/michael-dawson-6051282>



Overview

- What?
- Why?
- IT Self Service Agent - Laptop Refresh
- Using Deepeval to run Evaluations
- Cost
- CI/CD

What ?

- Tests to systematically measure AI system's performance in the context of a set of **goals**:
 - Accuracy
 - Reliability
 - Safety
- Don't necessary compare against specific expected outputs
- **Variation** allowed
 - what matters is if goals were met

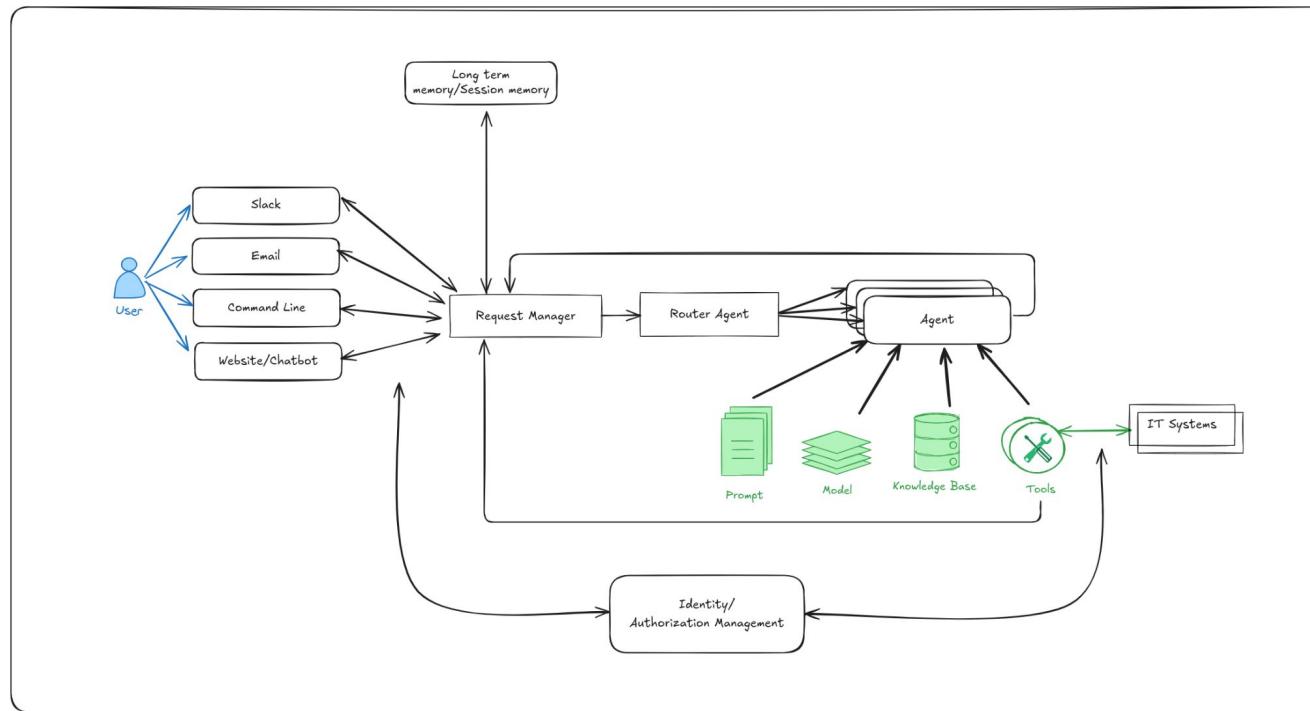
Why ?

- Evaluations are as critical as the agents themselves.
- Enables confident, rapid iteration on agent capabilities.
- Must be automated



- Of course we know that for existing systems
- It's just harder for agentic systems

Context - Self-service agent - laptop refresh



- Laptop policy RAG database
- Service now MCP

- Blog post series <https://developers.redhat.com/articles/2026/01/26/ai-quickstart-self-service-agent-it-process-automation>
- GitHub repository [rh-ai-quickstart/it-self-service-agent](https://github.com/rh-ai-quickstart/it-self-service-agent)

The stages we are progressing through

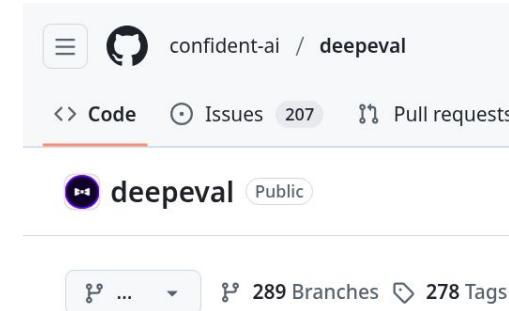
- Stage 1: Manual testing with a few predefined conversations
- Stage 2: Automated evaluation with basic metrics
- Stage 3: Use case specific metrics with context
- Stage 4: Multi-type coverage (predefined + generated + known bad)
- Stage 5: CI/CD integration
- Stage 6: Cost-awareness (**We are here**)
- Stage 7: Additional context from OTEL integration + user feedback
- Stage 8: Continuous monitoring

Top 10 practices (based on our experience so far)

1. Embrace Non-Determinism
2. Agree on "Good Enough" early
3. Build in Evaluation Hooks Early
4. Plan for Use-Case Specific Metrics
5. Plan for a Capable Evaluator Model
6. Augment Evaluations with Key Data
7. Test Your Tests (Known Bad Cases)
8. Automate in CI/CD from the Start
9. Establish Baselines and Track Regression
10. Track and Manage Token Usage Costs

Deepeval - Why ?

- Asked across internal slack channels, no recommended framework
 - Prompt foo mentioned
 - but not as strong recommendation
 - Did not seem to cover multi-turn conversations
- Why choose DeepEval ?
 - Multi-turn
 - Showed up in a number of searches
 - Seemed active
 - Licence - Apache 2
- <https://github.com/confident-ai/deepeval>



Deeper Dive

A conversation

Hello! I'm the routing agent, and I'm here to help you get connected with the right specialist agent to assist you with your query. My role is to quickly understand your needs and direct you to the most suitable expert who can provide you with the best possible support.

What brings you here today? Could you please describe what you need help with? We currently have specialist agents available to assist with laptop refresh requests and email address updates. Is either of these topics related to your query, or is there something else you need assistance with? I'm all ears and ready to help!

[Turn 1]  USER

I need help with laptop refresh

[Turn 2]  ASSISTANT

Here is your laptop information:

Employee Name: Isabella Mueller

Employee Location: EMEA

Laptop Model: ThinkPad T14

Laptop Serial Number: TP14006

Laptop Purchase Date: 2019-11-18

Laptop Age: 6 years and 1 month

Laptop Warranty Expiry Date: 2022-11-18

Laptop Warranty: Expired

Your laptop, a ThinkPad T14, was purchased on 2019-11-18 and is currently 6 years and 1 month old (73 total months). The company's laptop refresh policy states that standard laptops will be refreshed every 3 years from the date of issuance. Since your laptop is 6 years old (73 months), you are eligible for a laptop refresh. Would you like to proceed with reviewing the available laptop options for your location?

A conversation

[Turn 3]  USER

Yes, I'd like to proceed with reviewing the available laptop options for my location, please show me the list.

[Turn 4]  ASSISTANT

Here are the available laptop options for your location (EMEA):

1. Model: MacBook Air M2 - ServiceNow Code: apple_mac_book_air_m_2
 - Manufacturer: Apple
 - Target User: General Office
 - Cost: €1,299
 - Operating System: macOS
 - Display Size: 13.6 inches
 - Display Resolution: 2560 x 1664
 - Graphics Card: Apple M2 GPU
 - Minimum Storage: 256 GB SSD
 - Weight: 2.7 lbs
 - Ports: 2 x Thunderbolt/USB 4, MagSafe
 - Minimum Processor: Apple M2
 - Minimum Memory: 8 GB
 - Dimensions: 11.97 x 8.46 x 0.44 inches
- + other options

A conversation

[Turn 5]  USER

I'd like to select option 1, the MacBook Air M2 with ServiceNow Code: apple_mac_book_air_m_2, as it seems suitable for my general office needs.

[Turn 6]  ASSISTANT

You've selected the MacBook Air M2 with ServiceNow Code: apple_mac_book_air_m_2. Would you like to proceed with the creation of a ServiceNow ticket for a laptop refresh?

[Turn 7]  USER

Yes, please proceed with creating the ServiceNow ticket for the laptop refresh.

[Turn 8]  ASSISTANT

A ServiceNow ticket for a laptop refresh has been created for you. The ticket number is REQ3520696. Your request will be processed by the IT Hardware Team, and you will receive updates via email as the ticket progresses.
Is there anything else I can help you with?

Some Challenges

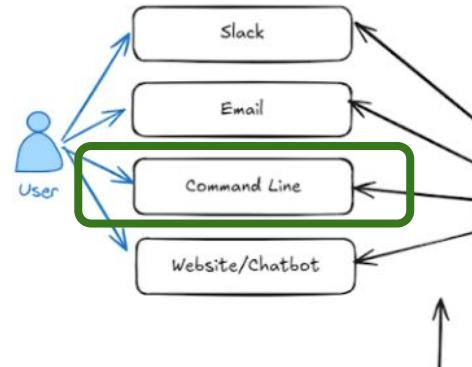
- Non-determinism
 - Conversation variation
 - Evaluation pickiness
 - Model variation
- Prompts extremely sensitive to changes
 - Subtle changes hard to spot manually
 - 1 conversation is not enough to validate
- Easy to over fit

Key components of Evaluations implementation

- Driving a conversation
- Generating conversations
 - Pre-defined flow
 - Generated flow
 - Known bad
- Evaluating conversations
- Full run
- Cost
- CI/CD

Driving a conversation

- Started with CLI interface



- oc exec

```
# Test with different user (LATAM region)
oc exec -it deploy/self-service-agent-request-manager -n $NAMESPACE -- \
    python test/chat-responses-request-mgr.py \
    --user-id maria.garcia@company.com
```

Driving a conversation

- .../evaluations/helpers/openshift_chat_client.py

```
cmd = [
    "oc", "exec", "-it", "deploy/self-service-agent-request-manager",
    "--",
    "bash",
    "-c",
    f"{env_vars} /app/.venv/bin/python {'/app/test/' + self.test_script if not self.test_script.startswith('/')
else self.test_script}",
]
self.process = subprocess.Popen(
    cmd,
    stdin=subprocess.PIPE,
    stdout=subprocess.PIPE,
    stderr=subprocess.PIPE,
    text=True,
    bufsize=1,
    universal_newlines=True,
)
```

Driving a conversation

openshift_chat_client.py

- **start_session()** - Start an interactive session with the chat client
- **get_agent_initialization()** - Wait for and capture the agent's initialization message
- **send_message(message, timeout=None)** - Send a message to the agent and get the response
- **get_app_tokens()** - Get the app tokens collected from the session
- **close_session()** - Close the chat session



Pre-defined conversations

Pre-defined conversations

```
.../evaluations/conversation_config/conversations:  
  return-to-router-done.json  
  return-to-router.json  
  success-flow-1.json  
  success-flow-2.json
```

Pre-defined conversations

```
{  
    "metadata": {  
        "authoritative_user_id": "alice.johnson@company.com",  
        "description": "Successful laptop refresh flow"  
    },  
    "conversation": [  
        {  
            "role": "user",  
            "content": "refresh"  
        },  
        {  
            "role": "user",  
            "content": "I would like to see the options"  
        },  
        {  
            "role": "user",  
            "content": "3"  
        },  
        {  
            "role": "user",  
            "content": "proceed"  
        }  
    ]  
}
```

[success-flow-1.json](#)

Pre-defined conversations

```
...
{
  "role": "user",
  "content": "no"
},
{
  "role": "user",
  "content": "DONEDONE DONE"
}
]
```

[return-to-router-done.json](#)

Pre-defined conversations

[.../evaluations/run_conversations.py](#)

An Example Conversation

.../evaluations/results/conversation_results/success-flow-1.json

```
{  
  "metadata": {  
    "authoritative_user_id": "alice.johnson@company.com",  
    "description": "Test results from success-flow-1.json"  
  },  
  "conversation": [  
    {  
      "role": "assistant",  
      "content": "Hello! I'm the routing agent, and I'm here to help you get connected with the right specialist agent to assist you with your query. My role is to quickly understand your needs and direct you to the most suitable expert who can provide you with the best possible support.\nWhat brings you here today? Could you please describe what you need help with? We currently have specialist agents available to assist with laptop refresh requests and email address updates. Is either of these topics related to your query, or is there something else you need assistance with? I'm all ears and ready to help!"  
    },  
    {  
      "role": "user",  
      "content": "refresh"  
    },  
    {  
      "role": "assistant",  
      "content": "Here is your laptop information:\nEmployee Name: Alice Johnson\nEmployee ID: 1001\nEmployee Location: EMEA\nLaptop Model: Latitude 7420\nLaptop Serial Number: DL7420001\nLaptop Purchase Date: 2020-01-15\nLaptop Age: 5 years and 11 months\nLaptop Warranty Expiry Date: 2023-01-15\nLaptop Warranty: Expired\nYour laptop, a Latitude 7420, was purchased on 2020-01-15 and is currently 5 years and 11 months old (71 total months). The company's laptop refresh policy states that standard laptops will be refreshed every 3 years from the date of issuance. Since your laptop is 5 years or older (71 months >= 36 months), you are eligible for a laptop refresh. Would you like to proceed with reviewing the available laptop options for your location?"  
    },  
  ]  
}
```

An Example Conversation

```
{  
    "role": "assistant",  
    "content": "Here are the available laptop options for your location (EMEA):\n1. Model: MacBook Air M2 - ServiceNow  
Code: apple_mac_book_air_m_2\n- Manufacturer: Apple\n- Target User: General Office\n- Cost: €1,299\n- Operating System: macOS\n- Display Size: 13.6 inches\n- Display Resolution: 2560 x 1664\n- Graphics Card: Apple M2 GPU\n- Minimum Storage: 256 GB SSD\n- Weight: 2.7 lbs\n- Ports: 2 x Thunderbolt/USB 4, MagSafe\n- Minimum Processor: Apple M2\n- Minimum Memory: 8 GB\n- Dimensions: 11.97 x 8.46 x 0.44 inches\n2. Model: MacBook Pro 16 M3 Max - ServiceNow Code:  
apple_mac_book_pro_16_m_3_max\n- Manufacturer: Apple\n- Target User: Developers\n- Cost: €3,299\n- Operating System: macOS\n- Display Size: 16.2 inches\n- Display Resolution: 3456 x 2234\n- Graphics Card: Apple M3 Max GPU\n- Minimum Storage: 1 TB SSD\n- Weight: 4.8 lbs\n- Ports: 3 x Thunderbolt 4, HDMI, SDXC, MagSafe\n- Minimum Processor: Apple M3 Max\n- Minimum Memory: 36 GB\n- Dimensions: 14.01 x 9.77 x 0.66 inches\n3. Model: ThinkPad T14s Gen 5 AMD - ServiceNow  
Code: lenovo_think_pad_t_14_s_gen_5_amd\n- Manufacturer: Lenovo\n- Target User: General Office\n- Cost: €1,380\n- Operating System: Windows 11 Pro\n- Display Size: 14 inches\n- Display Resolution: 1920 x 1200\n- Graphics Card: Integrated AMD Radeon\n- Minimum Storage: 512 GB SSD\n- Weight: 2.8 lbs\n- Ports: 2 x USB-C, 2 x USB-A, HDMI, Audio Jack\n- Minimum Processor: AMD Ryzen 7 PRO 7840U\n- Minimum Memory: 16 GB\n- Dimensions: 12.4 x 8.7 x 0.6 inches\n4. Model: ThinkPad P16 Gen 2 - ServiceNow Code: lenovo_think_pad_p_16_gen_2\n- Manufacturer: Lenovo\n- Target User: Engineers\n- Cost: €2,650\n- Operating System: Red Hat Linux\n- Display Size: 16 inches\n- Display Resolution: 2560 x 1600\n- Graphics Card: NVIDIA RTX A2000\n- Minimum Storage: 1 TB SSD\n- Weight: 6.5 lbs\n- Ports: 2 x USB-C, 2 x USB-A, HDMI, Ethernet, Audio Jack\n- Minimum Processor: Intel Core i7-13700HX\n- Minimum Memory: 32 GB\n- Dimensions: 14.3 x 10.5 x 1.2 inches\nPlease select one of the above options. Remember to consider your job role and requirements when making your selection."  
,  
{  
    "role": "user",  
    "content": "3"  
},
```

An Example Conversation

```
{  
    "role": "assistant",  
    "content": "You've selected the ThinkPad T14s Gen 5 AMD. Would you like to proceed with the creation of a  
ServiceNow ticket for a laptop refresh?"  
,  
{  
    "role": "user",  
    "content": "proceed"  
,  
{  
    "role": "assistant",  
    "content": "A ServiceNow ticket for a laptop refresh has been created for you. The ticket number is REQ2512134.  
Your request will be processed by the IT Hardware Team, and you will receive updates via email as the ticket  
progresses.\nIs there anything else I can help you with?"  
}  
]  
}
```

Generated Conversations - DeepEval Simulator

Generated Conversations - DeepEval Simulator

<https://github.com/rh-ai-quickstart/it-self-service-agent/blob/main/evaluations/generator.py>

```
from deepeval.simulator import ConversationSimulator
```

```
...  
simulator = ConversationSimulator(  
    model_callback=worker_model_callback,  
    simulator_model=custom_llm,  
)
```

```
...  
async def worker_model_callback(  
    input: str, turns: List[Turn], thread_id: str  
) -> Turn:  
    ...  
    worker_client = OpenShiftChatClient  
    ...  
    response = worker_client.send_message(input)  
    ...
```

Generated Conversations - DeepEval Simulator

```
# Create conversation golden
conversation_golden = _create_conversation_golden(conversation_number)

# Simulate conversation
conversational_test_cases = simulator.simulate(
    conversational_goldens=[conversation_golden],
    max_user_simulations=max_turns,
)

def _create_conversation_golden(conversation_number: int) -> ConversationalGolden:
    """Create a single ConversationalGolden object for simulation."""

    conversation_golden = ConversationalGolden(
        scenario="An Employee wants to refresh their laptop. The agent shows them a list they can choose from, they select the appropriate laptop and a Service now ticket number is returned.",
        expected_outcome="They get a Service now ticket number for their refresh request",
        user_description="user who tries to answer the assistants last question",
    )

    return conversation_golden
```

Generated Conversations - DeepEval Simulator

```
# Save test cases
for test_case in conversational_test_cases:
    test_case_count += 1
    conversation = _convert_test_case_to_conversation_format(
        test_case, conversation_user_id
    )
    if conversation.get("conversation"):
        base_filename = (
            f"generated_flow_worker{worker_id}_{test_case_count}"
        )
        saved_file = _save_conversation_to_file(conversation, base_filename)
        saved_files.append(saved_file)
```

results/conversation_results/generated_flow_worker0_1_20251219_161936.json

Generated Conversations - DeepEval Simulator

```
usage: generator.py [-h] [--max-turns MAX_TURNS] [--test-script TEST_SCRIPT]
                    [--reset-conversation] [--concurrency CONCURRENCY]
                    [--message-timeout MESSAGE_TIMEOUT]
                    [num_conversations]
```

alice.johnson@company.com

john.doe@company.com

maria.garcia@company.com

oliver.smith@company.com

yuki.tanaka@company.com

isabella.mueller@company.com

carlos.rodriguez@company.com

david.chen@company.com

sophie.dubois@company.com

ahmed.hassan@company.com

Known Bad Conversations

Known Bad Conversations

[.../evaluations/results/known_bad_conversation_results](#)

- allowed_invalid_user-laptop-selection.json
- bad_laptop_options.json
- did-not-confirm-service-now-creation.json
- fail-route-back-to-router.json
- incomplete.json
- missing-laptop-details.json
- missing_laptop_options2.json
- not-all-laptop-options.json
- no-ticket-number.json
- wrong_eligibility.json
- wrong-selection.json

python evaluate.py --check

Evaluating Conversations

Evaluating Conversations

```
.../evaluations/results/conversation_results
    ├── generated_flow_worker0_1_20251219_161936.json
    ├── known_good_flow.json
    ├── return-to-router-done.json
    ├── return-to-router.json
    ├── success-flow-1.json
    └── success-flow-2.json
```

Evaluating Conversations

[.../evaluations/get_deep_eval_metrics.py](#)

```
=====
RESULTS FOR: generated_flow_worker0_15_20260107_065430.json
=====
```

METRIC BREAKDOWN:

- ✓ PASS Turn Relevancy [Conversational GEval]: 1.000 (threshold: 0.8)
- ✓ PASS Role Adherence [Conversational GEval]: 1.000 (threshold: 0.5)
- ✓ PASS Conversation Completeness [Conversational GEval]: 1.000 (threshold: 0.8)
- ✓ PASS Information Gathering [Conversational GEval]: 1.000 (threshold: 0.8)
- ✓ PASS Policy Compliance [Conversational GEval]: 1.000 (threshold: 0.8)
- ✓ PASS Option Presentation [Conversational GEval]: 1.000 (threshold: 0.8)
- ✓ PASS Process Completion [Conversational GEval]: 1.000 (threshold: 0.8)
- ✓ PASS User Experience [Conversational GEval]: 1.000 (threshold: 0.8)
- ✓ PASS Flow termination [Conversational GEval]: 1.000 (threshold: 0.8)
- ✓ PASS Ticket number validation [Conversational GEval]: 1.000 (threshold: 1.0)
- ✓ PASS Correct eligibility validation [Conversational GEval]: 1.000 (threshold: 1.0)
- ✓ PASS No errors reported by agent [Conversational GEval]: 1.000 (threshold: 1.0)
- ✓ PASS Correct laptop options for user location [Conversational GEval]: 1.000 (threshold: 1.0)
- ✓ PASS Confirmation Before Ticket Creation [Conversational GEval]: 1.000 (threshold: 1.0)
- ✓ PASS Return to Router After Task Completion [Conversational GEval]: 1.000 (threshold: 1.0)

 PASS RATE: 15/15 (100.0%)

Evaluating Conversations

```
ConversationalGEval(
```

```
    name="Correct eligibility validation",
```

```
    threshold=1.0,
```

```
    model=custom_model,
```

```
    evaluation_params=[TurnParams.CONTENT, TurnParams.ROLE],
```

```
    evaluation_steps=[
```

"Look for any reference to the laptop refresh policy timeframe in the conversation. The agent may state this in multiple ways:",

- " - Direct statement: 'laptops will be refreshed every 3 years' or 'standard laptops are refreshed every 3 years'",

- " - Indirect via eligibility: 'your laptop is over 3 years old and eligible' (implies 3-year policy)",

- " - Indirect via ineligibility: 'your laptop is 2 years old and not yet eligible' (implies 3-year policy)",

- " - Reference to age threshold: 'laptops older than 3 years' or 'laptops less than 3 years old'"

"If the agent mentions ANY timeframe (whether directly or indirectly through eligibility statements), verify it is consistent with the policy in the additional context below, which states that standard laptops are refreshed every 3 years.",

"If the agent does NOT mention any timeframe or eligibility age at all, this evaluation PASSES (the metric only validates correctness when stated, not whether it's stated).",

"The user's eligibility status (eligible or not eligible) is irrelevant - only the accuracy of the stated or implied timeframe matters.",

```
f"\n\nadditional-context-start\n{default_context}\nadditional-context-end",
```

```
],  
,
```

Additional Context

Evaluating Conversations

```
ConversationalGEval(  
    name="Confirmation Before Ticket Creation",  
    threshold=1.0,  
    model=custom_model,  
    evaluation_params=[TurnParams.CONTENT, TurnParams.ROLE],  
    evaluation_steps=[
```

"Identify where in the conversation the user selects a laptop (e.g., selecting option '3', saying 'I'll go with the MacBook', etc.).",

"Identify where in the conversation a ServiceNow ticket is created (look for ticket numbers like 'REQ' followed by numbers, or statements like 'A ServiceNow ticket has been created').",

"Between the laptop selection and ticket creation, verify that:",

" a) The agent explicitly asks the user for confirmation to proceed with ticket creation (e.g., 'Would you like to proceed with creating a ServiceNow ticket?', 'Shall I create the ticket?', 'Would you like me to submit this request?', etc.)",

" b) The user has an opportunity to respond with their confirmation (e.g., 'proceed', 'yes', 'go ahead', etc.)",

" c) The ticket creation happens AFTER the user confirms, not before",

"If the ticket is created immediately after laptop selection without the agent first asking for confirmation and waiting for user response, this evaluation FAILS.",

"Note: The confirmation question must come from the agent BEFORE the ticket is created. If the agent creates the ticket and then asks 'Is there anything else I can help you with?', this does NOT count as confirmation - the ticket was already created.",

```
],  
)
```

Evaluating Conversations - DeepEval

[.../evaluations/deep_eval.py](#)

```
custom_model = CustomLLM(  
    api_key=api_key_found, base_url=current_endpoint or "", model_name=model_name  
)  
...  
# Discover all conversation files to evaluate  
json_files = [f for f in os.listdir(results_dir) if f.endswith(".json")]  
  
metrics = get_metrics(  
    custom_model, validate_full_laptop_details=validate_full_laptop_details  
)  
  
# Process each conversation file individually  
for filename in json_files:  
    ...  
    conversation_data = file_data["conversation"]  
    authoritative_user_id = file_data.get("metadata", {}).get(  
        "authoritative_user_id"  
)  
  
    # Convert to turns  
    turns = _convert_to_turns(conversation_data)
```

Evaluating Conversations - DeepEval

```
# Build test case context starting with conversation identifier  
test_case_context = [f'Laptop refresh conversation from {filename}']  
  
# Set chatbot role  
chatbot_role = """You are an IT Support Agent specializing in hardware replacement.
```

Your responsibilities:

1. Determine if the authenticated user's laptop is eligible for replacement based on company policy
2. Clearly communicate the eligibility status and policy reasons to the user
3. If the user is NOT eligible:
 - Inform them of their ineligibility with the policy reason (e.g., laptop age)
 - Provide clear, factual information that proceeding may require additional approvals or be rejected
 - Allow them to continue with the laptop selection process if they choose to
4. Guide the user through laptop selection
5. After the user selects a laptop, ALWAYS ask for explicit confirmation before creating the ServiceNow ticket (e.g., "Would you like to proceed with creating a ServiceNow ticket for this laptop?")
6. Only create the ServiceNow ticket AFTER the user confirms they want to proceed
7. After creating the ticket, provide the ticket number and next steps
8. Maintain a professional, helpful, and informative tone throughout

Note: Providing clear, factual information about potential rejection or additional approvals is sufficient. You do not need to be overly cautionary or repeatedly emphasize warnings. Always confirm with the user before creating tickets."""

Evaluating Conversations - DeepEval

```
test_case = ConversationalTestCase(  
    turns=turns,  
    context=test_case_context if test_case_context else None, # type: ignore[arg-type]  
    chatbot_role=chatbot_role,  
)  
  
# Execute evaluation with suppressed output for cleaner reporting  
display_config = DisplayConfig(print_results=False)  
results = evaluate(  
    test_cases=[test_case], metrics=metrics, display_config=display_config  
)  
  
# Store results with basic tracking for summary  
conversation_result: dict[str, Any] = {  
    "filename": filename,  
    "conversation_turns": len(turns),  
    "evaluation_results": results,  
    "metrics": [],  
    "total_metrics": len(metrics),  
}
```

Evaluating Conversations - DeepEval

[.../evaluations/results/deep_eval_results/deepeval_generated_flow_worker2_1_20251128_133338.json](#)

```
{  
  "metric": "Correct eligibility validation [Conversational GEval]",  
  "score": 1.0,  
  "success": true,  
  "reason": "The conversation fully meets the criteria as the agent correctly states the laptop refresh policy  
timeframe, mentioning that standard laptops will be refreshed every 3 years from the date of issuance, and the  
user's laptop is eligible for refresh since it is 4 years and 9 months old, aligning with the policy stated in the  
additional context.",  
  "threshold": 1.0,  
  "retry_performed": false  
},
```

Evaluating Conversations - DeepEval

```
=====
RESULTS FOR: generated_flow_worker0_15_20260107_065430.json
=====
```

METRIC BREAKDOWN:

- ✓ PASS Turn Relevancy [Conversational GEval]: 1.000 (threshold: 0.8)
- ✓ PASS Role Adherence [Conversational GEval]: 1.000 (threshold: 0.5)
- ✓ PASS Conversation Completeness [Conversational GEval]: 1.000 (threshold: 0.8)
- ✓ PASS Information Gathering [Conversational GEval]: 1.000 (threshold: 0.8)
- ✓ PASS Policy Compliance [Conversational GEval]: 1.000 (threshold: 0.8)
- ✓ PASS Option Presentation [Conversational GEval]: 1.000 (threshold: 0.8)
- ✓ PASS Process Completion [Conversational GEval]: 1.000 (threshold: 0.8)
- ✓ PASS User Experience [Conversational GEval]: 1.000 (threshold: 0.8)
- ✓ PASS Flow termination [Conversational GEval]: 1.000 (threshold: 0.8)
- ✓ PASS Ticket number validation [Conversational GEval]: 1.000 (threshold: 1.0)
- ✓ PASS Correct eligibility validation [Conversational GEval]: 1.000 (threshold: 1.0)
- ✓ PASS No errors reported by agent [Conversational GEval]: 1.000 (threshold: 1.0)
- ✓ PASS Correct laptop options for user location [Conversational GEval]: 1.000 (threshold: 1.0)
- ✓ PASS Confirmation Before Ticket Creation [Conversational GEval]: 1.000 (threshold: 1.0)
- ✓ PASS Return to Router After Task Completion [Conversational GEval]: 1.000 (threshold: 1.0)

✗ PASS RATE: 15/15 (100.0%)

CONVERSATION SUMMARY:

- ✓ generated_flow_worker0_2_20260107_064143.json: 15/15 metrics passed
- ✓ generated_flow_worker0_17_20260107_065622.json: 15/15 metrics passed
- ✓ generated_flow_worker0_15_20260107_065430.json: 15/15 metrics passed
- ✓ generated_flow_worker0_8_20260107_064745.json: 15/15 metrics passed
- ✓ generated_flow_worker0_9_20260107_064844.json: 15/15 metrics passed
- ✓ generated_flow_worker0_13_20260107_065236.json: 15/15 metrics passed
- ✓ success-flow-1.json: 15/15 metrics passed
- ✓ generated_flow_worker0_10_20260107_064942.json: 15/15 metrics passed
- ✓ generated_flow_worker0_11_20260107_065038.json: 15/15 metrics passed
- ✓ generated_flow_worker0_19_20260107_065816.json: 15/15 metrics passed
- ✓ generated_flow_worker0_3_20260107_064239.json: 15/15 metrics passed
- ✓ generated_flow_worker0_5_20260107_064437.json: 15/15 metrics passed
- ✓ generated_flow_worker0_16_20260107_065526.json: 15/15 metrics passed
- ✓ generated_flow_worker0_4_20260107_064338.json: 15/15 metrics passed
- ✓ generated_flow_worker0_20_20260107_065913.json: 15/15 metrics passed
- ✓ return-to-router-done.json: 15/15 metrics passed
- ✓ return-to-router.json: 15/15 metrics passed
- ✓ generated_flow_worker0_12_20260107_065135.json: 15/15 metrics passed
- ✓ generated_flow_worker0_7_20260107_064646.json: 15/15 metrics passed
- ✓ success-flow-2.json: 15/15 metrics passed
- ✓ known_good_flow.json: 15/15 metrics passed
- ✓ generated_flow_worker0_18_20260107_065719.json: 15/15 metrics passed
- ✓ generated_flow_worker0_14_20260107_065333.json: 15/15 metrics passed
- ✓ generated_flow_worker0_6_20260107_064552.json: 15/15 metrics passed
- ✓ generated_flow_worker0_1_20260107_064046.json: 15/15 metrics passed

🎉 OVERALL RESULT: ALL CONVERSATIONS PASSED

```
=====
```

Evaluating Conversations - DeepEval

```
usage: deep_eval.py [-h] [--api-endpoint API_ENDPOINT] [--api-key API_KEY]
                    [--results-dir RESULTS_DIR] [--output-dir OUTPUT_DIR]
                    [--context-dir CONTEXT_DIR] [--validate-full-laptop-details]
                    [--no-validate-full-laptop-details]
```

- Will generally need a “strong” model as judge
- Llama3 70b, “just good enough in our quickstart”

Full run

1. evaluate.py (Orchestrator)
↓
2. Cleanup Phase
 - Remove previous generated conversations
 - Clear old token usage files
↓
3. run_conversations.py
 - Execute predefined conversations
 - Save results to results/conversation_results/
↓
4. generator.py
 - Generate synthetic conversations
 - Add to results/conversation_results/
↓
5. deep_eval.py
 - Evaluate ALL conversations (predefined + generated)
 - Apply comprehensive metrics
 - Generate reports in results/deep_eval_results/
↓
6. Results Aggregation
 - Combine token usage statistics
 - Generate summary reports
 - Calculate overall pass rates

- Known good flows
- Known bad flows
- Generated flows

Cost

📱 App Tokens (from chat agents):

Input tokens: 631,018

Output tokens: 31,680

Total tokens: 662,698

API calls: 154

Max single request input: 0

Max single request output: 0

Max single request total: 0

💡 Evaluation Tokens (from evaluation LLM calls):

Input tokens: 1,096,793

Output tokens: 33,056

Total tokens: 1,129,849

API calls: 555

Max single request input: 5,754

Max single request output: 143

Max single request total: 5,827

Using standard rates for one of the frontier models:

Input tokens: 1,600,000 tokens ×
\$0.30 per million = \$0.48

Output tokens: 64,000 tokens ×
\$2.50 per million = \$0.16

Total Cost: \$0.64

CI/CD

**Evaluations + CI/CD
Critical to achieving reasonable velocity**

CI/CD

Makefile

```
.PHONY: test-short-resp-integration-request-mgr
```

```
test-short-resp-integration-request-mgr:
```

```
    @echo "Running short responses integration test with Request Manager..."
```

```
    uv --directory evaluations run evaluate.py -n 1 --test-script chat-responses-request-mgr.py --reset-conversation  
$(VALIDATE_LAPTOP_DETAILS_FLAG)
```

```
    @echo "short responses integrations tests with Request Manager completed successfully!"
```

```
.PHONY: test-long-resp-integration-request-mgr
```

```
test-long-resp-integration-request-mgr:
```

```
    @echo "Running long responses integration test with Request Manager..."
```

```
    uv --directory evaluations run evaluate.py -n 20 --test-script chat-responses-request-mgr.py --reset-conversation  
--timeout=1800 $(VALIDATE_LAPTOP_DETAILS_FLAG)
```

```
    @echo "long responses integrations tests with Request Manager completed successfully!"
```

```
.PHONY: test-long-concurrent-integration-request-mgr
```

```
test-long-concurrent-integration-request-mgr:
```

```
    @echo "Running long concurrent responses integration test with Request Manager..."
```

```
    uv --directory evaluations run evaluate.py -n 10 --test-script chat-responses-request-mgr.py --reset-conversation  
--timeout=1800 --concurrency 4 --message-timeout 120 $(VALIDATE_LAPTOP_DETAILS_FLAG)
```

```
    @echo "long concurrent responses integrations tests with Request Manager completed successfully!"
```

CI/CD

 **All checks have passed** ^

3 successful checks

-  Pull Request - Integration + End to End Tests / Run e2e tests (pull_r...) Required ...
-  Pull Request - Quality Checks & Unit Tests / Code Quality Check (pul... Required ...
-  Pull Request - Quality Checks & Unit Tests / Run unit-tests (pull_req... Required ...

CI/CD

All workflows

Showing runs from all workflows

Filter workflow runs

3,023 workflow runs

Event ▾ Status ▾ Branch ▾ Actor ▾

✓ (dev) Prod deploy - Nightly 20 conversation test

(dev) Prod deploy - Nightly 20 conversation test #22: Scheduled

main

Today at 1:34 AM

28m 50s

...

✓ (dev) (Scout Prompt) Nightly 20 conversation test

(dev) (Scout Prompt) Nightly 20 conversation test #20: Scheduled

main

Today at 12:31 AM

27m 40s

...

✓ (dev) (Small Prompt) Nightly 20 conversation test

(dev) (Small Prompt) Nightly 20 conversation test #20: Scheduled

main

Jan 6, 11:30 PM EST

33m 13s

...

✓ (dev) Pull Request - Nightly 20 conversation test

(dev) Pull Request - Nightly 20 conversation test #20: Scheduled

main

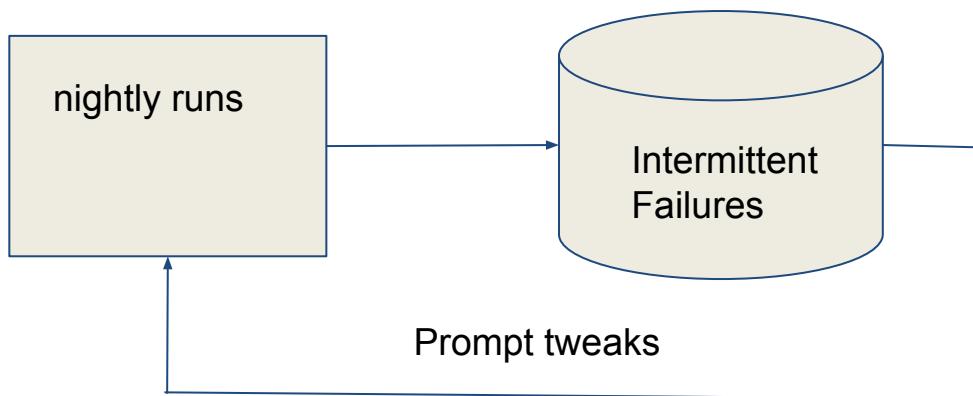
Jan 6, 11:00 PM EST

31m 8s

...

CI/CD

Incremental improvement



```
(evaluations) midawson@midawson-virtualbox:~/newpull/self-service-agent-blueprint$ claude
Claude Code v2.1.1
Sonnet 4.5 · API Usage Billing
~/newpull/self-service-agent-blueprint

/model to try Opus 4.5. Note: you may need to request access from your cloud provider

> /review-nightly
```

13/12/25	N/A	N/A	1 False Positive	N/A
14/12/25	N/A	N/A	1 False Positive	N/A
15/12/25	N/A	N/A	1 False Positive 1 Server error '500 Internal Serv	N/A

Recap

- What?
- Why?
- IT Self Service Agent - Laptop Refresh
- Using DeepEval to run Evaluations
- Cost
- CI/CD

Thank You

Questions?



Copyright and Trademarks

© Red Hat, IBM. All Rights Reserved

Red Hat, the Red Hat logos are trademarks or registered trademarks of Red Hat

IBM, the IBM logo, ibm.com are trademarks or registered trademarks of International Business Machines Corp.,

registered in many jurisdictions worldwide.

A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at

www.ibm.com/legal/copytrade.shtml

Node.js is an official trademark of Joyent. IBM SDK for Node.js is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

Java, JavaScript and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

npm is a trademark of npm, Inc.

Other trademarks or logos are owned by their respective owners.