

Using AI Agents to Automate Networking Script Development

NANOG 93

Randy Zhang, Principal Architect, Cisco Systems

Feb 2025

Agenda

1. Introduction
2. Agentic AI for Script Development
3. Building an Automation Framework
4. An Example Walkthrough with Autogen
5. Summary

Introduction

Problem Statement

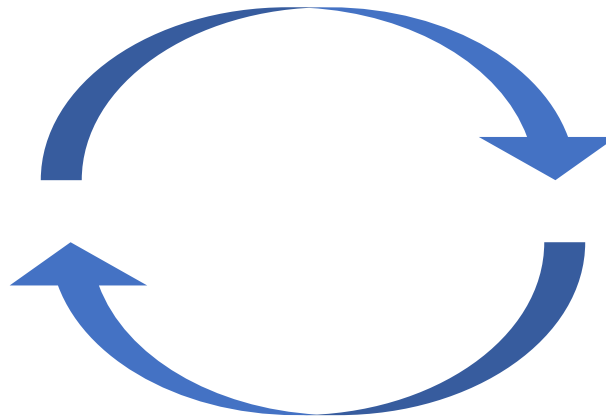
How can we leverage AI's generative capabilities to automate networking script development?

Networking Script Development Lifecycle

Planning and
defining
requirements

Documentation
and reporting

Preparing device
access and instructions



Script
development

Script testing with
live devices

Bug fixes and
enhancement

Need for Accelerated Development

- Frequent changes in requirements
- Compressed timeline
- Automated testing
- Reusable code (for future use) to disposable code (created on-demand)?

**Agent based Generative AI (Agentic AI)
presents a new low-code alternative for
network engineers**

What Can We Demand from AI?

1. Understand my networking requirements and convert them into an action plan for coding
2. Write code based on the plan
3. Review and improve the code
4. Execute the code and fix errors
5. Report on what is done
6. Validate the code with real networking hardware

The result is a development lifecycle leading to a working script that is validated

Agentic AI for Script Development

Introduction to Agentic AI

What is Agentic AI?

AI systems designed to **autonomously** pursue complex goals and **workflows** with limited direct human supervision. It is sometimes called Autonomous AI.

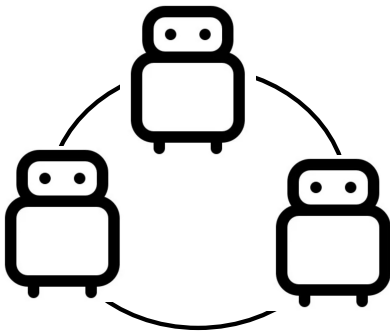
Components of Agentic AI



Agents

- An autonomous worker unit that builds on LLM
- It can make decisions independently
- An agent can perform one or more tasks
- Agents can collaborate and delegate tasks to other agents (to form an agency or a crew)
- Agents can form a workflow to iterate

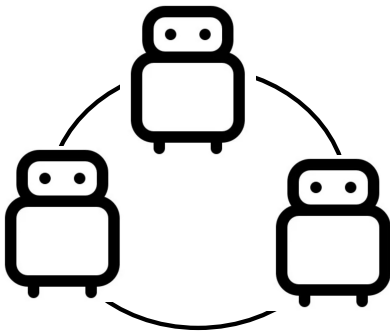
Agent Functions



- **Planning**: breaking down complex tasks into manageable steps, determining sequence of actions, self prompting
- **Reflection**: reviewing and critiquing their own outputs, iterate based on self feedbacks
- **Tool use**: using specialized tools to finish specific tasks
- **Collaboration**: Delegating work to other agents

Agent Enhances Foundational AI Models

Limitations of Foundational AI Models*



- Static and stale knowledge
- Designed for Q and A
- May hallucinate and generate authentically false info
- Need careful prompting and context

* These are quickly changing with the newer reasoning models (that use reasoning tokens)

Agentic AI vs Standalone AI Applications

Autonomous AI Applications

Collaboration and Decision Making

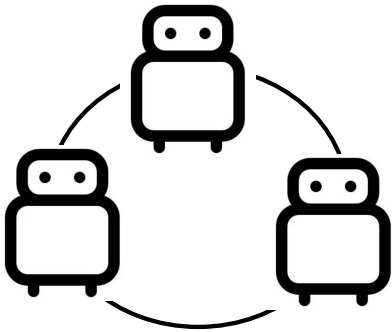
- Goal based instructions with little or no human interventions
- Can make decisions and take actions autonomously

Standalone AI Applications

Specific Capabilities and Use Cases

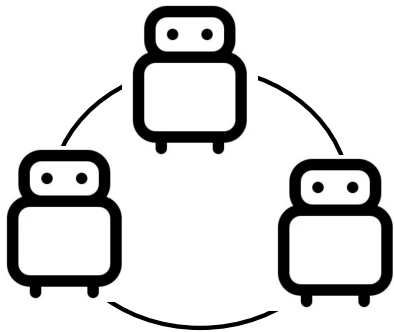
- Designed for human to machine interaction (eg chatbot)
- Requires careful prompting or continued intervention

Agentic AI for Networking Script Development



- Multi-agent framework to divide and conquer a complex software development project
- Iterations to improve the quality of the code
- Automated testing and validation with actual networking equipment
- Mixture of foundational models can be leveraged for each model's strengths

Agent Instruction Example for a Networking Script



You lead a team to collaborate on a Python networking script development project. The final completed Python script should perform the following 3 tasks:

1. Collect device running configuration for BGP, and loopback interfaces
2. Compare the BGP running configuration of each device to best practices
3. Summarize compliance and make detailed recommendations

A Sample Agent Configuration

Agent name

Message to agent for its role

```
Planner = ConversableAgent(  
    name="Planner",  
    system_message="""You are a software architect.  
You generate clear and detail coding instructions per  
requirements for Coder.  
    You can generate low level logic to help Coder  
but you do not write the code""",  
    llm_config=llm_config,  
)
```


Building an Automation Framework

Building a Development Environment

1. Describe the problem

2. Setting up the environment

3. Test drive and improve

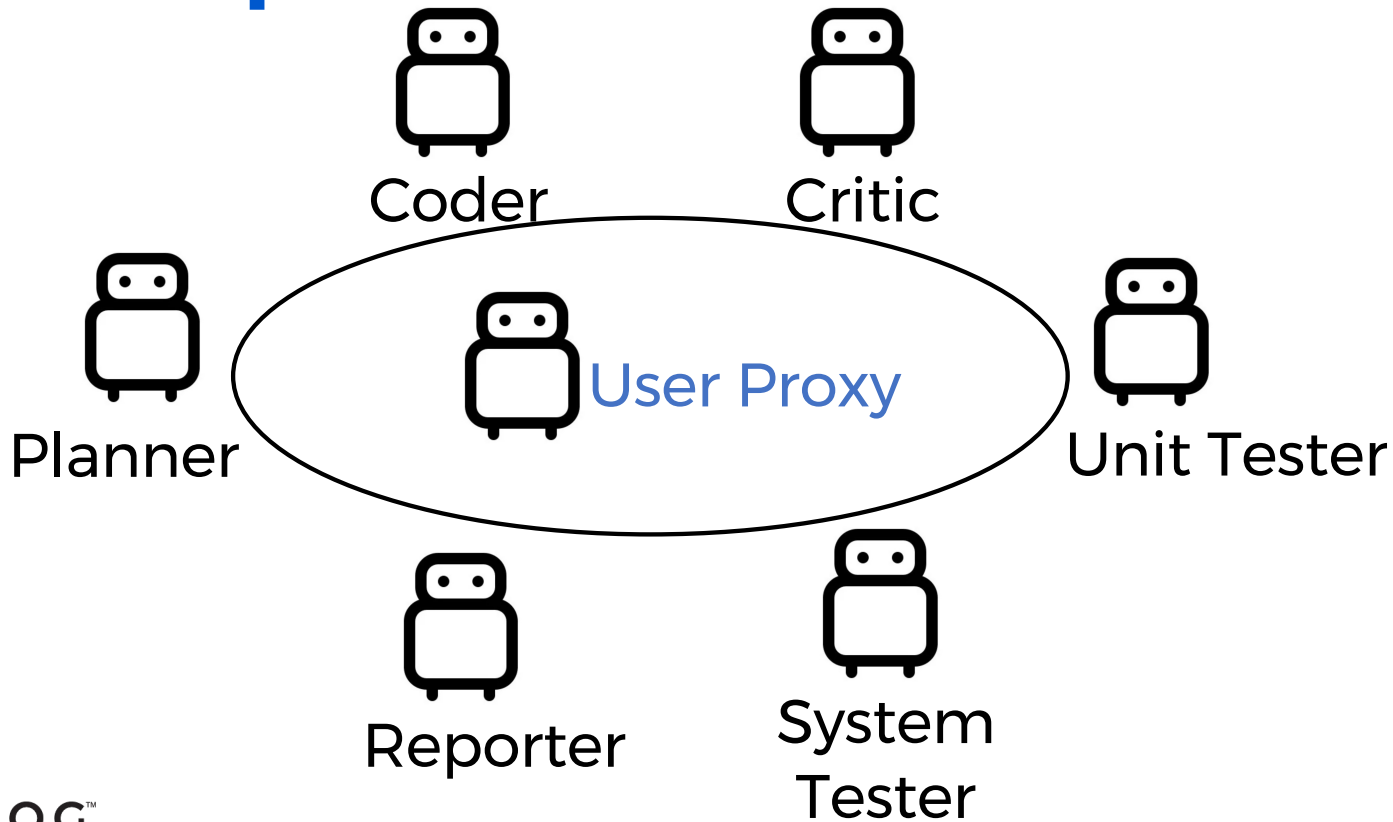


Building a Development Environment

1. Describe the problem

- What networking issues are you trying to solve (problem statement),
- Break up the problem into smaller units,
- Create a high level plan,
- Capture key requirements

Divide and Conquer with AI Agents: An Example



Agents Roles, an Example

Planner	Kicks off the project with a detailed project plan and role assignment
Coder	Writes and debugs code per requirements
Critic	Reviews and updates code and makes suggestions for enhancement
Unit Tester	Writes and runs unit tests
System Tester	Writes and runs system tests
Reporter	Summarizes the project and writes code documentation
User Proxy	Admin, user interaction

Building a Development Environment

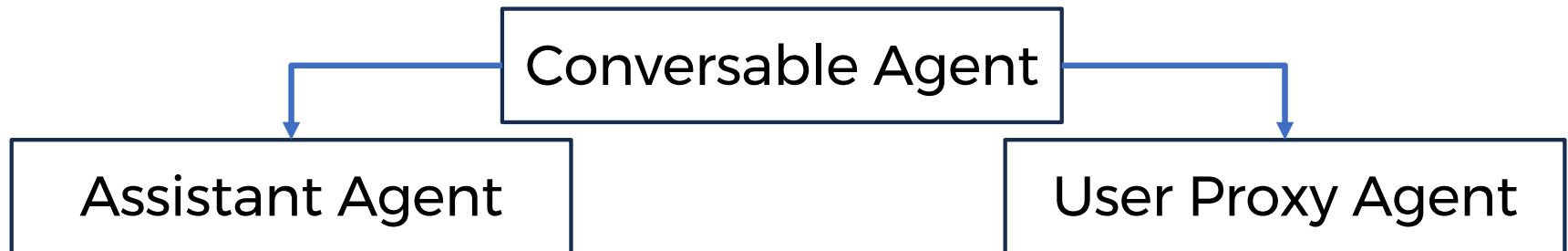
2. Setting up the environment

- Select an AI agent development framework,
- Select one or more foundational models (eg cost, context window size, rate limits, performance, hallucination rate),
- Get API keys,
- Identify which scripting language,
- Collect device access credentials for live testing

Microsoft Autogen Agent Framework

- Open-sourced multi-agent conversational framework
- Agents leverage Large Language Models (LLMs) for code generation and message exchanges
- Agents can receive and send messages, execute code, and specialize in different tasks.

Agent Templates



Building a Development Environment

3. Test drive and improve

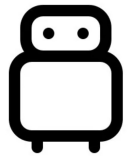
- Put all together to kickoff the project,
- Review the outputs
- Fine tune agents and system messages

An Example Walkthrough with Autogen

Project Description

1. Create a Python script to
 - + Collect router BGP configurations,
 - + Compare with BGP best practices,
 - + Summarize the compliance, and generate recommendations in a report.
2. Validate the script using actual routers.

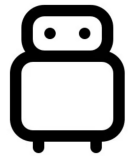
AI Agents



Planner



Coder



Critic



Reporter



User Proxy

Select Models

```
openai_config_list = [  
    {  
        "model": "gpt-4o",  
        "api_key": OPENAI_API_KEY,  
        "base_url": azure_endpoint,  
    },  
    {  
        "model": "gpt-4-turbo",  
        "api_key": OPENAI_API_KEY,  
        "base_url": azure_endpoint,  
    }  
]
```

Configure Models

```
llm_config_openai = {  
    "config_list": openai_config_list,  
}  
  
llm_config = llm_config_openai
```

Setting up Planner Agent

```
Planner = ConversableAgent(  
    name="Planner",  
    system_message="""You are a software  
architect. You generate clear and detail  
coding instructions per requirements for  
Coder.  
    You can generate low level logic to help  
Coder but you do not write the code""",  
    llm_config= llm_config,  
)
```

Setting up Coder Agent

```
Coder = AssistantAgent(  
    name="Coder",  
    system_message="""You are a Python  
developer. You generate code per instructions  
and fix errors in the code""",  
    llm_config= llm_config,  
)
```

Setting up Critic Agent

```
Critic = AssistantAgent(  
    name="Critic",  
    system_message="""You are a Python code  
reviewer and critic. You provide  
recommendations to improve the code""",  
    llm_config= llm_config,  
)
```


Setting up Reporter Agent

```
Reporter = AssistantAgent(  
    name="Reporter",  
    system_message="""You are a Python code  
reporter and documentor.  
    You write and publish documents about the  
code so that code is easier to maintain.  
    Your report should also include unit  
results and system test results""",  
    llm_config= llm_config,  
)
```

Setting up User Proxy Agent

```
user_proxy = UserProxyAgent(  
    name = "user_proxy",  
    human_input_mode = "ALWAYS",  
    code_execution_config={  
        # the executor to run the generated code  
        "executor": LocalCommandLineCodeExecutor(work_dir="tmp_dir"),  
    },  
)
```

Setting up Group Chats

```
group_chat = GroupChat(  
    agents = [user_proxy, Planner, Coder,  
Critic, Reporter],  
    messages = [],  
    send_introductions= True,  
    speaker_selection_method = "auto",)  
  
group_chat_manager = GroupChatManager(  
    groupchat = group_chat,  
    llm_config = llm_config,)
```

Initiating the Workflow

```
chat_result = user_proxy.initiate_chat(  
    group_chat_manager,  
    message=""" You lead a team to collaborate on a  
networking script development project using Python.  
The Python script should perform the following  
3 tasks:  
    <snip>      """,  
    summary_method="reflection_with_llm",  
    max_turns=5,  
)
```

A Sample Planner Chat (1/3)

Great, let's dive into the requirements and break them down into actionable steps for Coder to implement. Here's our plan:

Step-by-Step Plan

<snip>

Detailed Coding Instructions for Coder

<snip>

Here's the detailed logical structure and low-level logic for each step:

<snip>

A Sample Planner Chat (2/3)

Next Steps for Coder

1. Implement the script as per the plan.
2. Add additional best practice checks in ``compare_bgp_config``.
3. Execute the script and validate it works as intended.
4. Make necessary adjustments based on Critic and Reporter feedback.

A Sample Planner Chat (3/3)

Reporter

Ensure detailed documentation and how to run the script, along with sample output files, are included.

Once the script is fully implemented and documented, it will be ready for execution and validation. Let's proceed with coding!

Automated Error Fixing (1/2)

```
>>>>>>> EXECUTING CODE BLOCK (inferred language is python)...
user_proxy (to chat_manager):

exitcode: 1 (execution failed) Code output: File
"C:\zhang\proj\autogen_routers\tmp_dir\tmp_code_f50787a7c02f0bd4797
5b17ab7177fa9.py", line 44 'remove_private_as': 'remove-private-
AS', ^

SyntaxError: invalid syntax

-----
```


Automated Error Fixing (2/2)

Coder (to chat_manager):

It appears that there might be a syntax issue within the dictionary definition. Let's revise the code for clarity and syntax correction:

<snip>

>>>>>>> EXECUTING CODE BLOCK (inferred language is python)...
user_proxy (to chat_manager):

exitcode: 0 (execution succeeded)

Code output:

A Sample Reporter Chat (1/2)

Script Execution Report

The script executed successfully with an exit code of 0. Below is a detailed breakdown of the steps taken, observations, and the results generated.

Execution Summary:

- The script read the `routers.csv` file.
- It connected to each device listed in the file.
- Retrieved the BGP configurations.
- Compared the configurations against best practices.
- Generated recommendations for each device.
- Logged all significant events and errors.

A Sample Reporter Chat (2/2)

Logs:

The script generated a log file (`script.log`) detailing the execution. The contents of the log file are as follows:

<snip>

Output Files:

<snip>

Sample Configuration File (`router1_bgp.txt`):

<snip>

Recommendations (`recommendations.txt`):

<snip>

Conclusion:

The script performed its intended tasks successfully, generating configuration files and recommendations for each device. All significant events were logged, and no errors were encountered during execution.

Additional Info: Code Repositories



[Automated networking script development with AI agents](#)

[Project Neo: an AI-powered automated event-routing and response system](#)



Additional Info: White Papers



[Building AI Agents to Automate Enterprise-Level Software Development: A Practical Perspective](#)

[Demystifying AI Networking Protocols](#)



Summary

Observations

- Expect agents to go back and forth with multiple iterations to fix various coding problems.
- Examples of common Python coding problems that agents often create: improper indentation, incorrect module structure or import, stuck in a loop of recurring bugs
- Expect different results with each run, or with different foundational models
- Agents may push the LLM token usage beyond the limits

Conclusions

- Agentic AI presents a low-code approach to automate networking script development
- To produce quality code, you may need to provide clear messaging, careful code review, persistent coaching and tuning
- It's important to have a live testing environment so that the generated code can be automatically validated

Final Quote

The hottest new programming language is English.

Andrej Karpathy
co-founder of OpenAI



Thank you