

Launch of Iva IDE

From Prompt to Script – Instantly.

We are thrilled to launch Iva IDE, a cutting-edge solution powered by Generative AI, designed to accelerate automated script development through intuitive vibe coding. Iva IDE empowers users to rapidly build robust, reusable Brics with minimal coding effort. By embracing a citizen developer model, it eliminates the barrier of programming expertise—enabling anyone to create and deploy scripts with ease.

BENEFITS

- ▶ **Up to 90% reduction** in manual scripting effort through GenAI-based automation
- ▶ **70-80% faster automation** Compared to traditional workflows
- ▶ **Citizen developer – Zero/ Low code experience**
Leverage natural language to create scripts without writing a single line of code
- ▶ **Trained for 6+ scripting frameworks**
including Ansible, Terraform, Python, Shell, PowerShell, and REST APIs
- ▶ **Flex LLM**
Supports multiple LLMs; choose preferred model for script generation

IDE for Bric Creation

A GenAI-powered Web IDE for Automation and Catalog Builders

Iva IDE is an intuitive, web-based integrated development environment built specifically for creating, editing, and deploying Brics. Backed by a powerful coding LLM, it enables users to write and test automation scripts using natural language, reducing development time and complexity.

Key Workflow Steps



METADATA INPUT

Start by entering key details—Bric name, description, and Bric Type. This ensures clarity and discoverability across the platform.

The form contains the following fields and controls:

- Bric Name:** Text input with a character count '0 of 20'.
- Description:** Text input with a character count '0 of 300'.
- Image:** Image upload area with a 'Choose File' button and a note 'No file chosen'. Below it, a small text says 'Upload the image of the bric (256x146 px) in jpg, svg, png, or pdf format.' and a character count '0 of 450'.
- Release Notes:** Text input with a character count '0 of 450'.
- Select Bric:** Dropdown menu with 'Analisable Bric' selected.
- Select Provider:** Dropdown menu with 'Select an option'.
- Next:** Button to proceed.

GENERATING SCRIPT

In the IDE page, GenAI service generates the automation script based on the prompt provided. The prompt should include all the necessary details like dependencies required, error handling and specific inputs for variables if needed to reduce the need to refine the prompt.

```
1 - name: Install macOS software
2   hosts: localhost
3   gather_facts: false
4   tasks:
5     - name: Check if Homebrew is installed
6       command: /usr/bin/which brew
7       register: brew_check
8       changed_when: false
9       failed_when: brew_check.rc != 0
10      ignore_errors: yes
11
12     - name: Install Homebrew if not installed
13       shell: /bin/bash -c '$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)'
14       when: brew_check.rc != 0
15
16     - name: Update Homebrew
17       homebrew:
18         update_homebrew: yes
19
20     - name: Install a software package
21       homebrew:
22         name: '{{ software_name }}'
23         state: present
24         vars_prompt:
25           - name: 'software_name'
26             prompt: 'Enter the name of the software to install'
```

Key Points:

- **Homebrew:** The playbook ensures Homebrew is installed before proceeding.
- **Installation:** The Homebrew Ansible module is used for installing macOS software.
- **Error Handling:** The playbook checks for Homebrew installation, updates it, installs the software, and verifies installation. If verification fails, an error message is shown.

Output:

```
1 Please wait for a few minutes while the task gets executed at the automation platform
2 {"id":"82964151-d2a8-4e7f-b446-9a9341ac782","dxc_tenant_id":"","taskInputHopping":{"c7a7a
```

LIVE TESTING SCRIPT

Users can use the Run & Debug after generating script generated to test the script live and make as many edits as required to ensure that the script is fully functional before saving.

```
1 tasks:
2   - name: Check if Homebrew is installed
3     command: /usr/bin/which brew --version
4     register: brew_check
5     ignore_errors: yes
6   - name: Install Homebrew
7     shell: >
8       /bin/bash -c '$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)'
9     when: brew_check is failed
10    - name: Ensure Homebrew is up to date
11      homebrew:
12        update_homebrew: yes
13    - name: Install wget
14      homebrew:
15        name: wget
16        state: present
17    - name: Verify wget installation
18      command: wget --version
19      register: wget_version
20      failed_when: wget_version.rc != 0
21      debug:
22        msg: 'Installed wget version: {{ wget_version.stdout_lines[0] }}'
23    hosts: localhost
24    become: yes
```

Key Points:

- **Homebrew:** The playbook ensures Homebrew is installed before proceeding.
- **Installation:** The Homebrew Ansible module is used for installing macOS software.
- **Error Handling:** The playbook checks for Homebrew installation, updates it, installs the software, and verifies installation. If verification fails, an error message is shown.

Output:

```
1 Please wait for a few minutes while the task gets executed at the automation platform
2 {"id":"82964151-d2a8-4e7f-b446-9a9341ac782","dxc_tenant_id":"","taskInputHopping":{"c7a7a
```