



Ansible Automation  
Platform

# Best practices to deploy Ansible Execution Environments

Alex Dworjan

Associate Principal Specialist Solution Architect

Red Hat Ansible

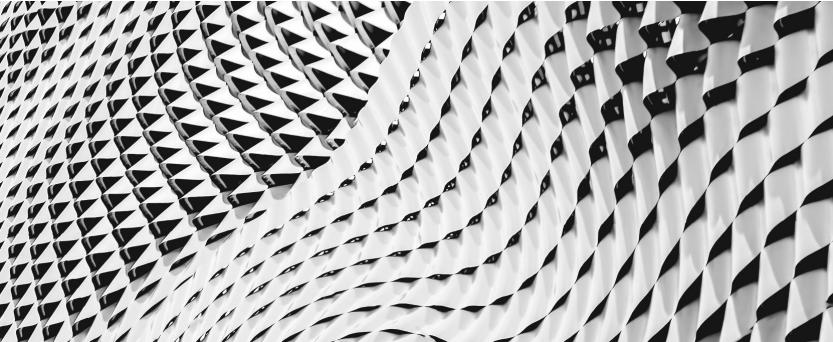
[www.youtube.com/@alexdworian](https://www.youtube.com/@alexdworian)



# What we'll discuss today

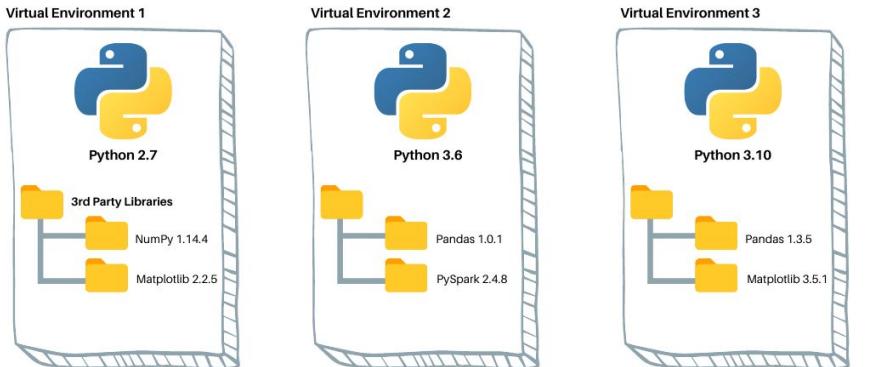
- ▶ What are execution environments and why do we need them.
- ▶ How to build a custom execution environment with Ansible
- ▶ Some guidelines

# What are Execution Environments



# Python Virtual environments

What did we have before?



There we specify the Python interpreter and that points to the folder with the third-party libraries installed.

Why are they important?

They create an isolated context for project's dependencies.

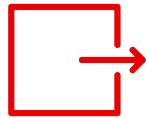
# Limitations of Python Virtual Environments

They don't work for the enterprise



## Tooling

Python Virtual Environments are not part of the Red Hat Ansible Automation Platform solution. These are Python constructs meant for Python developers.



## Portability

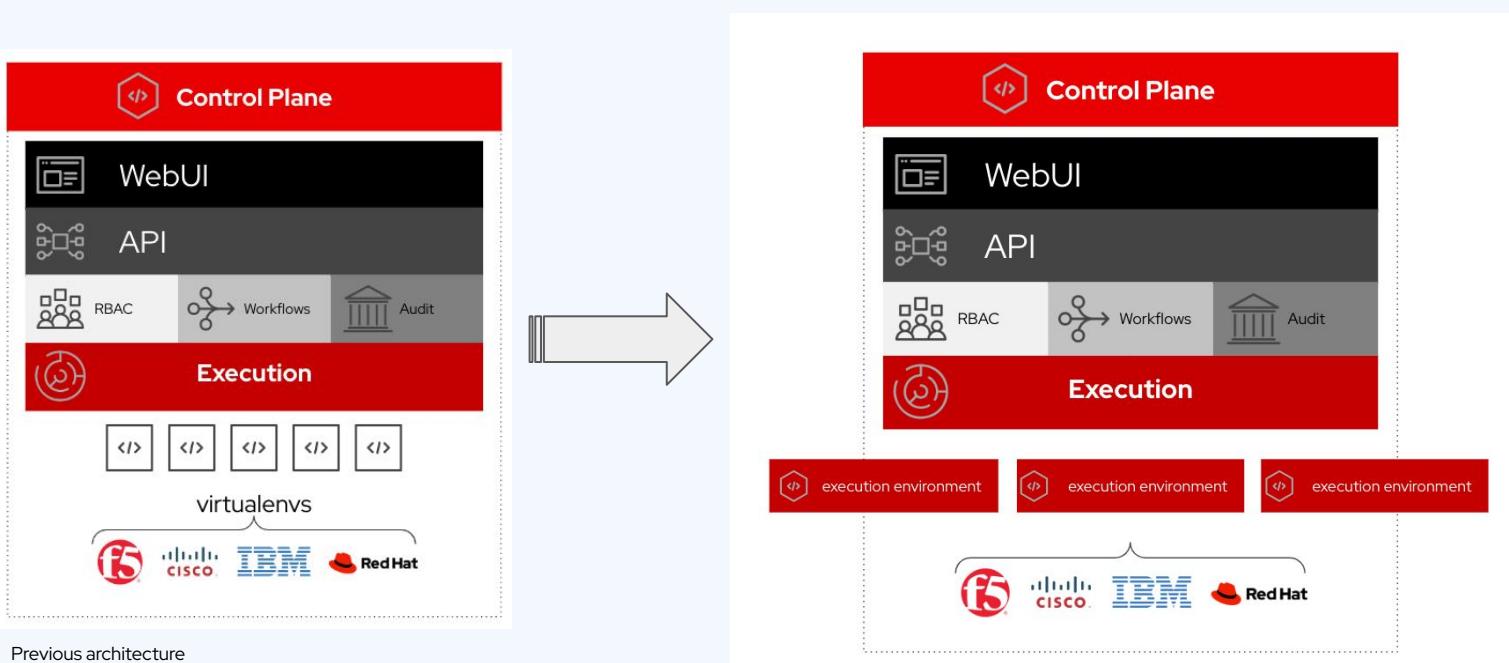
Python Virtual Environments are unique to a single system and hard to replicate on another system. It's difficult for one person to share their virtual environment with another person to replicate their automation.



## Maintenance

Python Virtual Environments may have dozens of Python dependencies per Ansible Automation Project and become increasingly hard to manage and maintain overtime.

# RedHat introduced EEs in AAP 2



# Virtual vs Execution Environments

## Main differences

### Virtual Environments

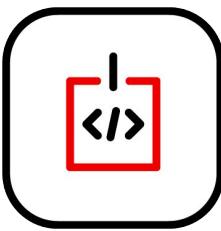
- ▶ Tightly coupled to the controller.
- ▶ Some commands run inside the venv, others run outside.
- ▶ Have to be created manually.

### Execution Environments

- ▶ As a container, can be run from any configured host or cluster.
- ▶ Will be decoupled from the controller which means horizontal scaling will be automatic.
- ▶ Entirely contained.
- ▶ Tools such as Ansible Builder will make creation easier.

# Automation Execution Environments

An automation execution environment is a container image used to execute Ansible playbooks and roles. It provides a defined, consistent, portable environment for executing automation.



Execution  
Environments



Collections



Libraries

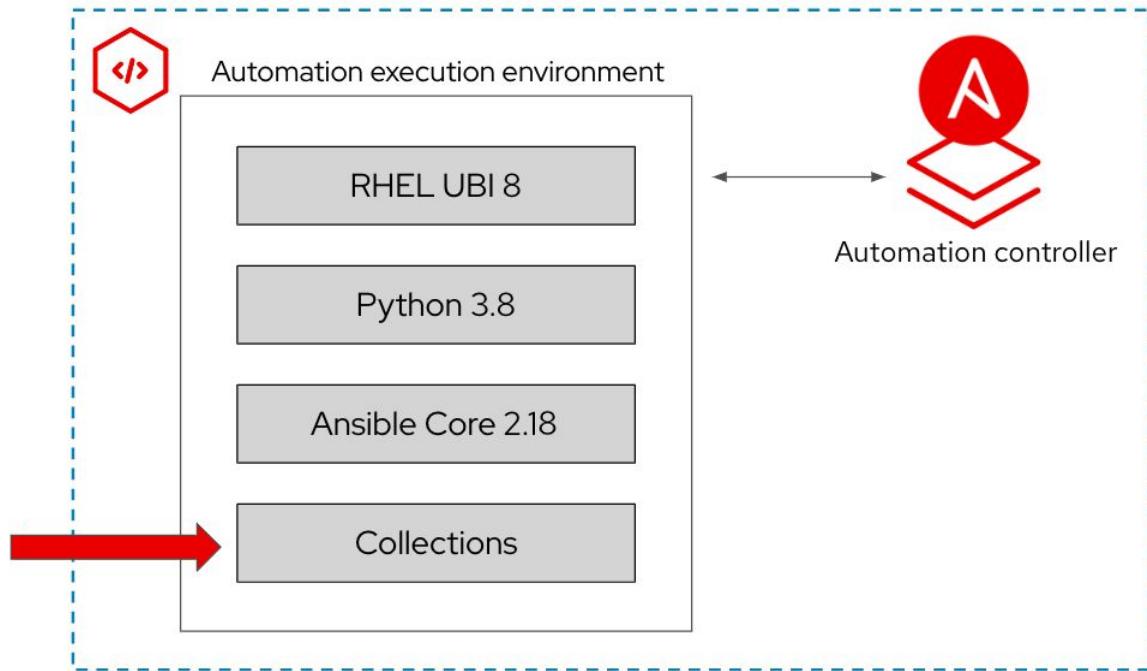


Ansible Core

Universal Base Image

# What's in an automation execution environment?

-  amazon.aws Collection
-  ansible.utils Collection
-  arista.cvp Collection
-  azure.azcollection Collection
-  ibm.qradar Collection
-  redhat.satellite Collection

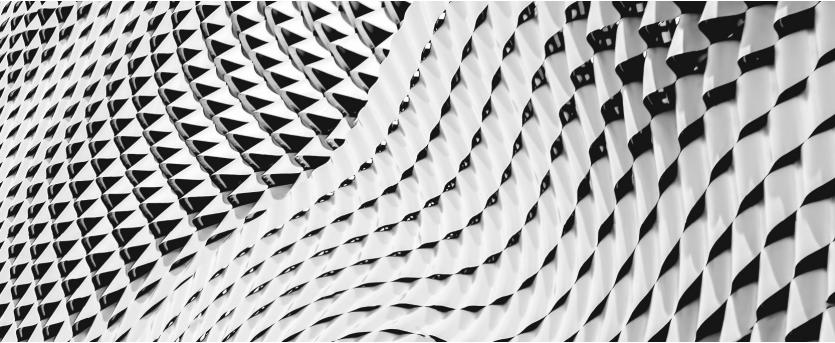


# Execution Environments available

With AAP 2.5

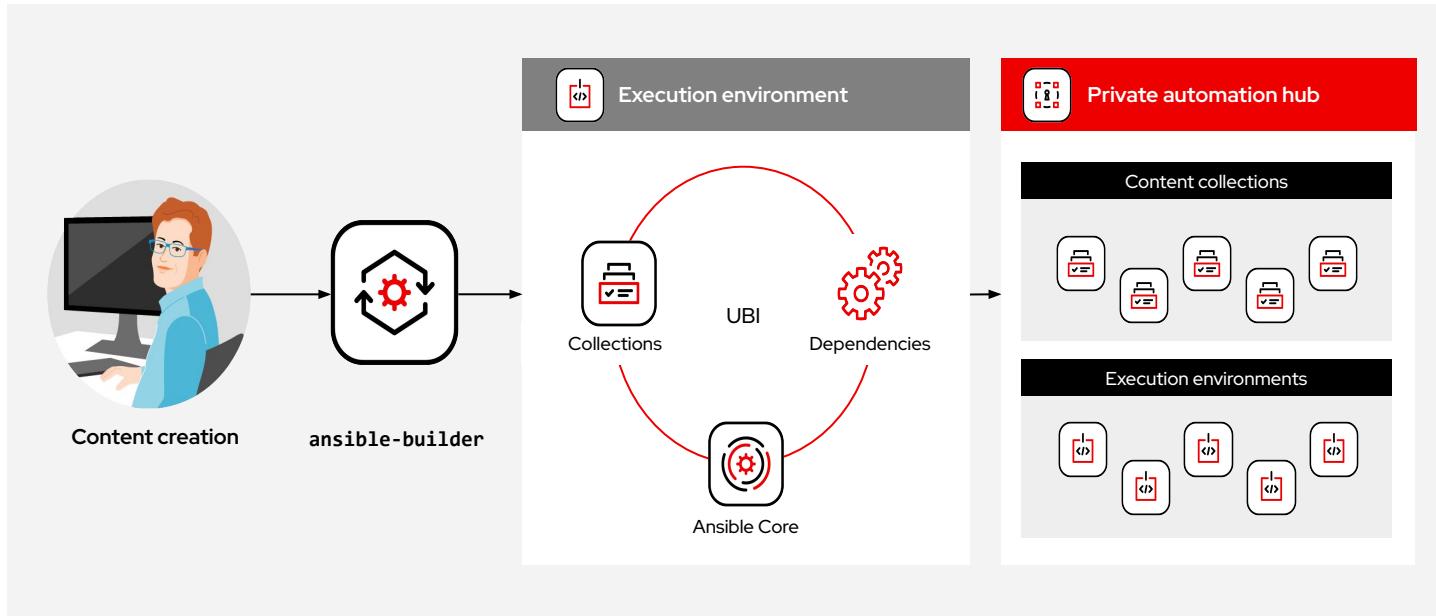
- ▶ **ee-minimal-rhel8**: minimal container image with ansible-core and basic collections.
- ▶ **ee-supported-rhel8** : container image with ansible-core and automation content collections supported by Red Hat.

# How to build an Execution Environment



# Execution environment development

Build, collaborate, sign, publish



# Key Sections

## Execution Environment Definition File

### **dependencies**

Dictionary value to define dependencies for Ansible Galaxy (collections or roles), Python (libraries) and operating system (packages).

### **options**

This section is a dictionary that contains keywords/options that can affect ansible-builder runtime functionality.

### **additional\_build\_files**

This section allows you to add any file to the build context directory. For example an ansible.cfg file to download collections from your private automation hub.

### **additional\_build\_steps**

This section enables you to specify custom build commands for any build phase.

# Definition file

## YAML Syntax

```
---
version: 3

images:
  base_image:
    name: registry.redhat.io/ansible-automation-platform-24/ee-minimal-rhel8:latest

dependencies:
  galaxy:
    collections:
      - ansible.netcommon

additional_build_files:
  - src: files # copy contents from files to configs dir in build context
    dest: configs

additional_build_steps:
  prepend_galaxy:
    # reference the file from build context like this
    - COPY _build/configs/ansible.cfg /etc/ansible/ansible.cfg

options:
  package_manager_path: /usr/bin/microdnf
```

# Dependencies

## Requirements files

requirements.yml

```
---  
collections:  
  - geerlingguy.java  
  - kubernetes.core
```

requirements.txt

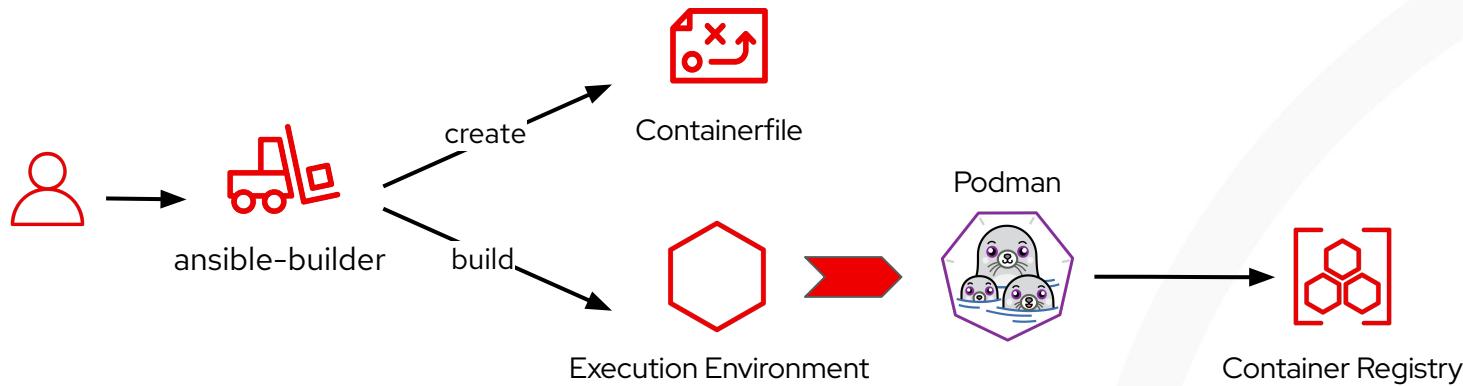
```
boto>=2.49.0  
botocore>=1.12.249  
pytz  
python-dateutil>=2.7.0  
awxkit  
packaging  
requests>=2.4.2  
xmltodict  
azure-cli-core==2.11.1  
python_version >= '2.7'  
collection community.vmware  
google-auth  
openshift>=0.6.2  
requests-oauthlib
```

bindep.txt

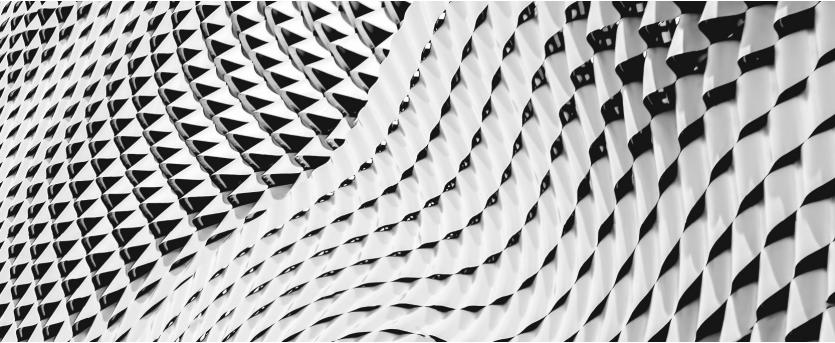
```
libxml2-devel [platform:rpm]  
subversion [platform:rpm]
```

# Execution Environment Building Workflow

## The Builder Tool



# Guidelines to build a EE



# How to build a good Execution Environment?

Apply the same approach as in software development, such as devops, even if it's a very simple project.

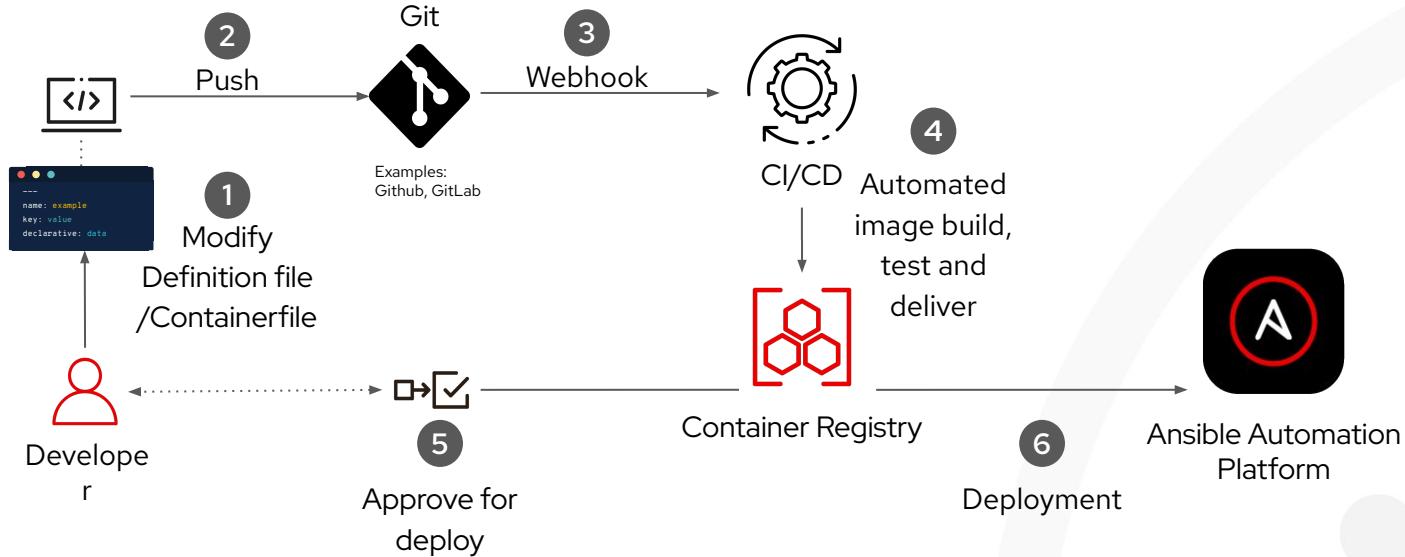
Build as many times as needed until all errors are fixed. Test changes and rollback in the event something needs to be addressed.

## 4 good habits for building EEs

1. Restrict images to only what is needed.
2. Update EEs regularly.
3. Do not create monoliths.
4. Keep track of versions and why you included pieces.

# Use Continuous Integration

## When building the Execution Environment



# EE as Code

What's the benefit?

- Version control history of every EE built kept in Source Control
- Provide access to modify EE definition without access to EE build host
- Provide Job Template to end-users to build new EE
  - Schedule job to rebuild EE
- Easy to build EE for new team / use-case



Requirements

- VM / Pipeline that includes podman / docker and ansible-builder
- Existing Execution environment with
  - infra.ee\_utilities
  - infra.ah\_configuration
  - ansible.controller
  - containers.podman or community.docker

## Steps to Achieve

Update Remotes  
requirements.yml



Update each collection remotes requirements.yml to ensure all necessary collections will be downloaded

Base EE Sync



Sync latest base EE to ensure latest ansible-core and vulnerabilities are addressed

Collection Repo  
Sync



Sync in all collection repos to ensure all necessary collections and versions are available

Build EE



Build the execution environment on a VM / via pipeline using the EE as code definition

Update controller

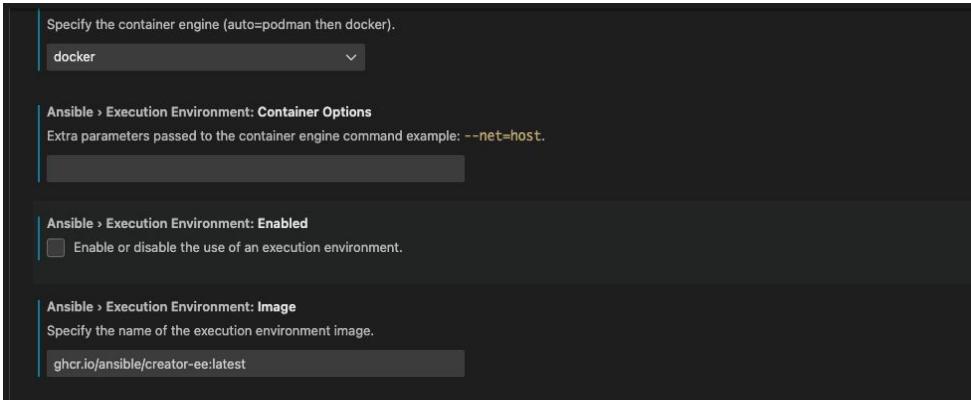


Update the Execution Environment tag in controller to ensure users utilize the correct version

# Ansible VSCode extension

Test syntax locally before pushing the definition file

- Point your vscode extension to creator-ee to get the built in ansible-core and ansible-lint support irrespective of your local environment



# Ansible Navigator

Test image locally before publishing

- ansible-navigator run test.yaml --eei custom-execution-environment:1.0

```
(base) asalgado@asalgado-mac ee_rsync_content % ansible-navigator run sync.yaml --eei custom-execution-environment:1.0 --pp missing -m stdout
[WARNING]: provided hosts list is empty, only localhost is available. Note that
the implicit localhost does not match 'all'

PLAY [localhost] ****
TASK [Gathering Facts] ****
ok: [localhost]

TASK [ansible.posix.synchronize] ****
fatal: [localhost]: FAILED! => {"changed": false, "cmd": "/usr/bin/rsync --delay-updates -F --compress --archive --out-format='<<CHANGED>>%i %n%L' /home/runner/some files/attrs were not transferred (see previous errors) (code 23) at main.c(1189) [sender=3.1.3]\n", "rc": 23}

PLAY RECAP ****
localhost : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
Please review the log for errors.
```

# Resources

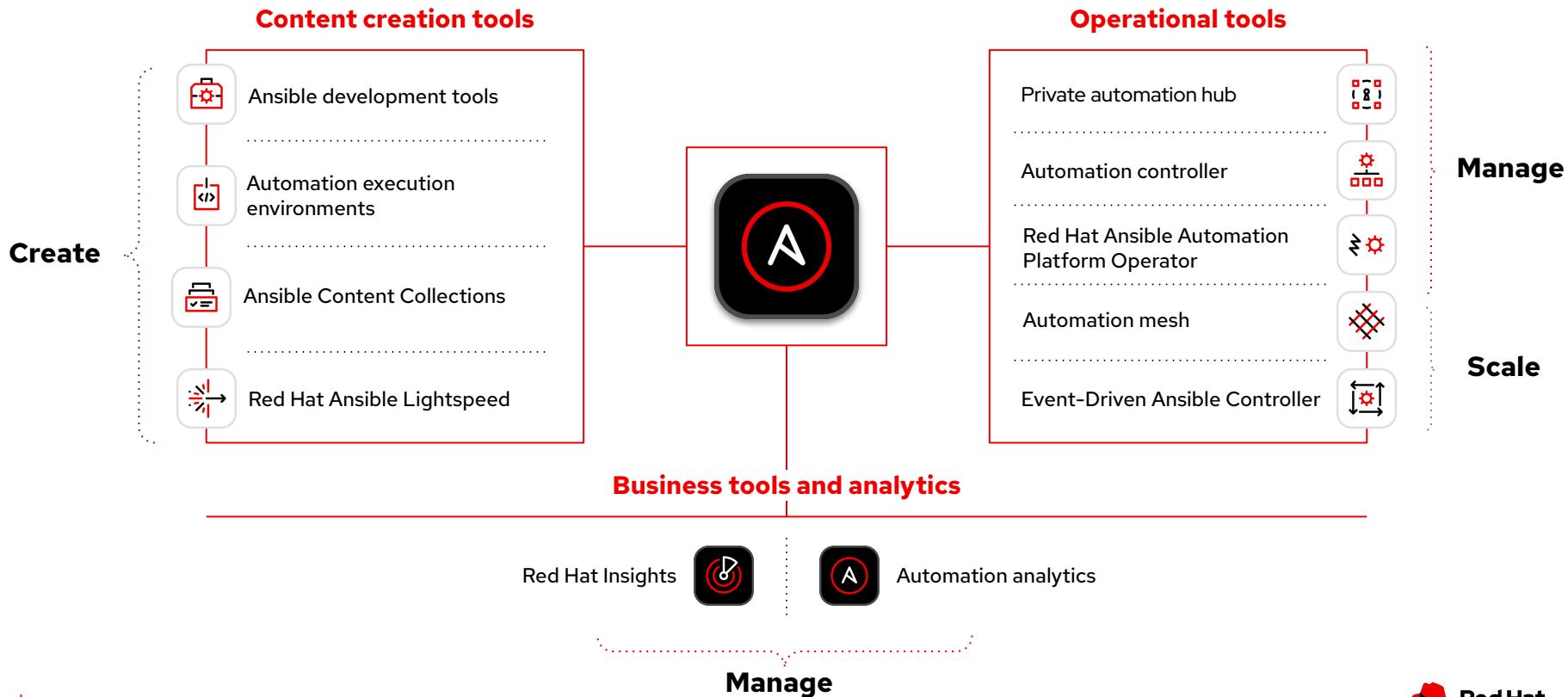
1. <https://www.ansible.com/blog/whats-new-in-ansible-automation-platform-2-automation-execution-environments>
2. <https://www.ansible.com/blog/unlocking-efficiency-harnessing-the-capabilities-of-ansible-builder-3.0>
3. <https://www.ansible.com/blog/automating-execution-environment-image-builds-with-github-actions>
4. <https://www.ansible.com/blog/the-anatomy-of-automation-execution-environments>
5. <https://www.redhat.com/architect/ansible-execution-environment-tips>
6. <https://www.redhat.com/architect/ansible-execution-environment-automated-build>
7. <https://www.ansible.com/blog/digitally-signing-ansible-content-collections-using-private-automation-hub>
8. <https://www.ansible.com/blog/project-signing-and-verification>
9. <https://www.redhat.com/en/interactive-labs/ansible>

# Red Hat Ansible Lightspeed

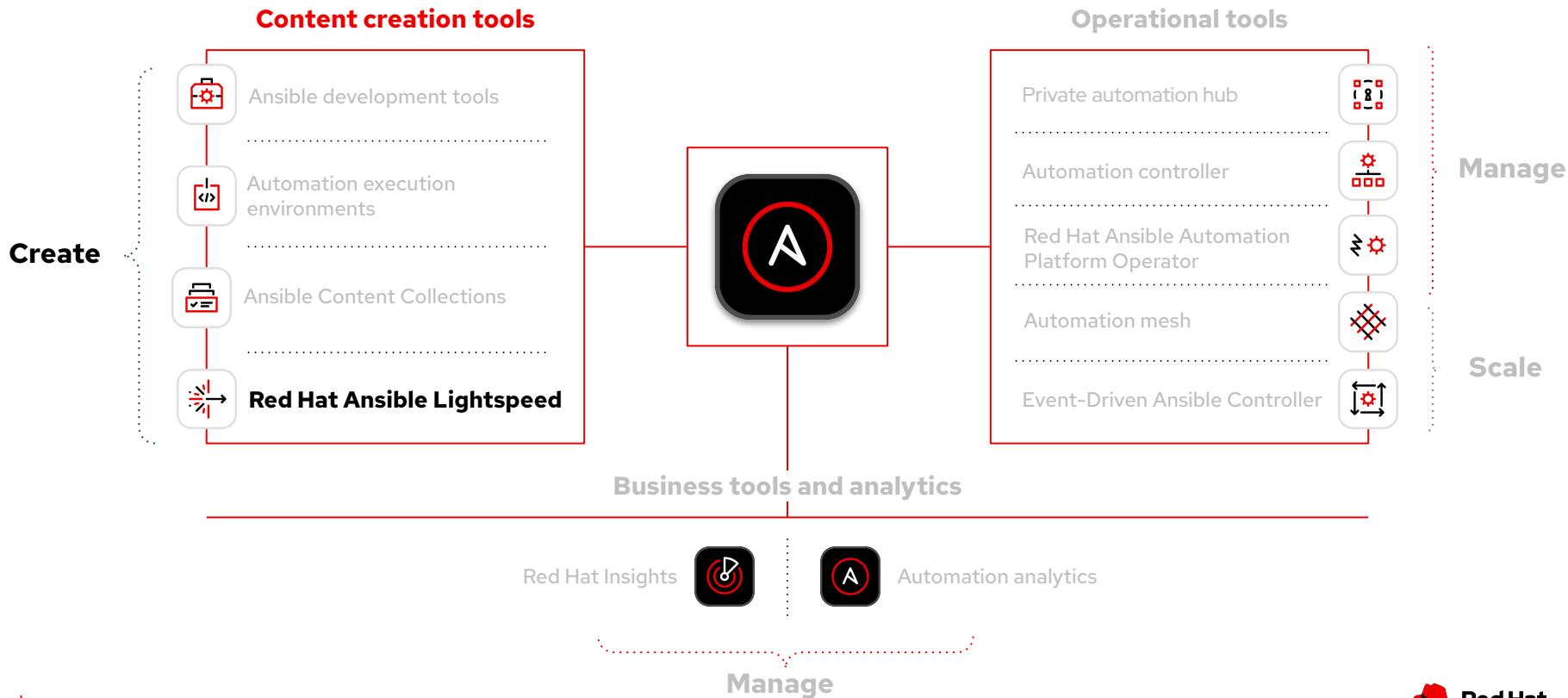
with IBM Watsonx Code Assistant



# Red Hat Ansible Automation Platform



# Red Hat Ansible Automation Platform





## Red Hat Ansible Lightspeed with IBM watsonx Code Assistant

### The developer interface

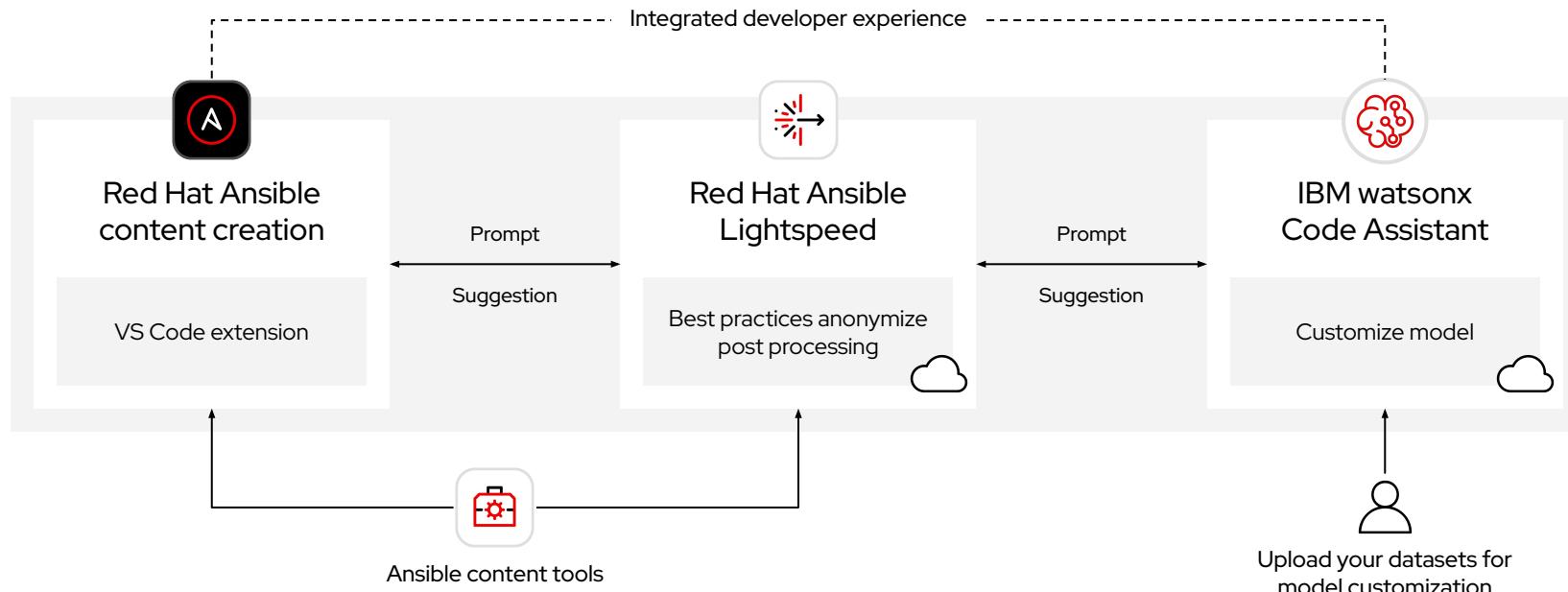
Deployed natively in Visual Studio Code via the Ansible extension

### The integrated service

Integration of AI services into Ansible Automation Platform with the Ansible VS Code extension

### The generative AI

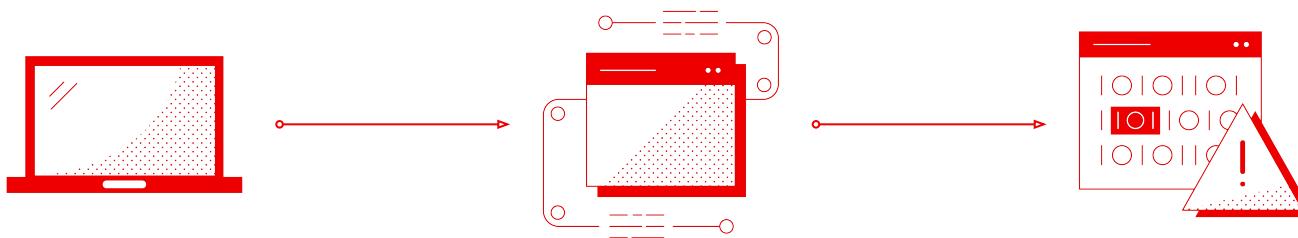
IBM watsonx Code Assistant is powered by the Ansible-specific watsonx.ai foundation model



### Data security

- + Prompt/Suggestion data is encrypted in transit and is ephemeral
- + Customer Ansible playbooks and customized models are stored in customer-owned cloud object storage and are not shared with IBM, Red Hat, or any other customers

# The automation code life cycle



## Create

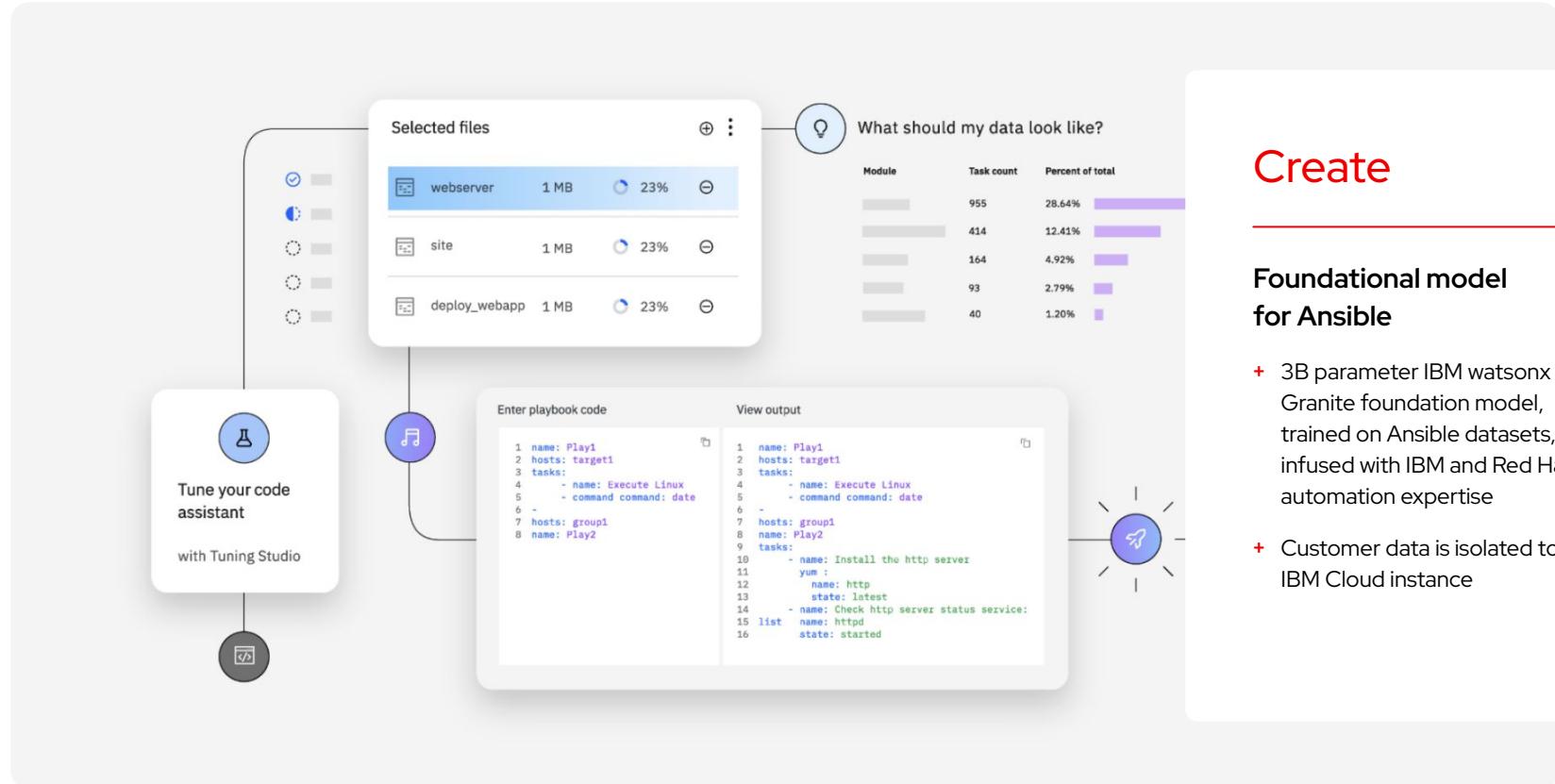
—  
Multi-task generation  
Foundational model for Ansible

## Adopt

—  
Content source matching  
Post-processing capabilities

## Maintain

—  
Ansible code bot  
User management



```
configure_postgres.yml •
playbooks > configure_postgres.yml
1 ---
2   - name: Configure Database servers
3     hosts: databases
4     become: true
5
6   tasks:
7     # Install postgresql-server & Run postgresql-setup command & Start and enable postgresql service & Allow the traffic
```

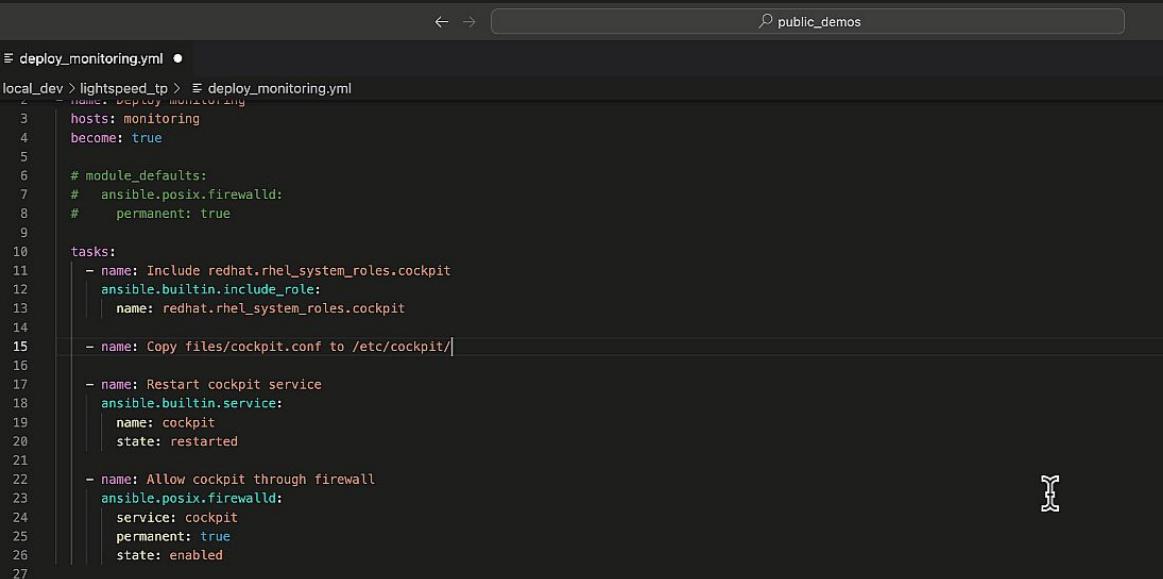
## Create

### Multi-task generation

Generates multiple task suggestions for Ansible task files and playbooks with a sequential chain of natural language task prompts chained together using ampersands (&) in YAML comment (#) lines

0 ▲ 0

Ln 7, Col 121 Spaces: 2 UTF-8 LF Ansible (EE) 2.15.3 Lightspeed Python 3.10.12



```
deploy_monitoring.yml •
local_dev > lightspeed_tp > deploy_monitoring.yml
  - name: Deploy monitoring
    hosts: monitoring
    become: true
    # module_defaults:
    #   ansible.posix.firewall:
    #     permanent: true
    tasks:
      - name: Include redhat.rhel_system_roles.ansible.builtin.include_role:
        name: redhat.rhel_system_roles.ansible.builtin.include_role:
          name: cockpit
      - name: Copy files/cockpit.conf to /etc/cockpit/
      - name: Restart cockpit service
        ansible.builtin.service:
          name: cockpit
          state: restarted
      - name: Allow cockpit through firewall
        ansible.posix.firewall:
          service: cockpit
          permanent: true
          state: enabled

```

## Adopt

### Content source matching

- + Will always attempt to clearly identify the potential sources of training content for maximum transparency and trust
- + Each generated code suggestion will provide a training source, author, and license

```
≡ deploy_monitoring.yml ×  
local_dev > lightspeed_tp > ≡ deploy_monitoring.yml  
1 ---  
2   - name: Deploy monitoring  
3     hosts: monitoring  
4     become: true  
5  
6     # module_defaults:  
7     #   ansible.posix.firewalld:  
8     #     permanent: true  
9  
10    tasks:  
11      # - name: Include redhat.rhel_system_roles.cockpit  
12  
13      # - name: Copy files/cockpit.conf to /etc/cockpit/  
14  
15      # - name: Restart cockpit service  
16  
17      # - name: Allow cockpit through firewall
```



## Adopt

### Post-processing capabilities

- + Ansible Lightspeed's post-processing capability augments model suggestions with Ansible best practices, subject matter expertise, and more
- + This processing and strong contextual awareness stems from Red Hat's unique insight into the Ansible code base, and proven experience helping customers automate at scale

### Training dataset creation

```
● (ansible-venv) ~ Ansible-OpenShift-Provisioning git:(main) ansible-content-parser https://github.com/IBM/Ansible-OpenShift-Provisioning.git /tmp/out2/ --skip-ansible-lint
INFO: pipeline:Running data pipeline
INFO: pipeline:Start scanning for 1 projects (total 269 files)
INFO: pipeline:Done
● (ansible-venv) ~ Ansible-OpenShift-Provisioning git:(main) ls /tmp/out2
ftdata.jsonl metadata report.txt repository
● (ansible-venv) ~ Ansible-OpenShift-Provisioning git:(main) ansible-content-parser . /tmp/out3/ --skip-ansible-lint
INFO: pipeline:Running data pipeline
INFO: pipeline:Start scanning for 1 projects (total 269 files)
INFO: pipeline:Done
○ (ansible-venv) ~ Ansible-OpenShift-Provisioning git:(main) █
```

### Model training

Tune a model

Provide the name and description of your tuning experiment. You can configure your tuned model within the experiment in the next step.

Define details

The base code model you are going to tune

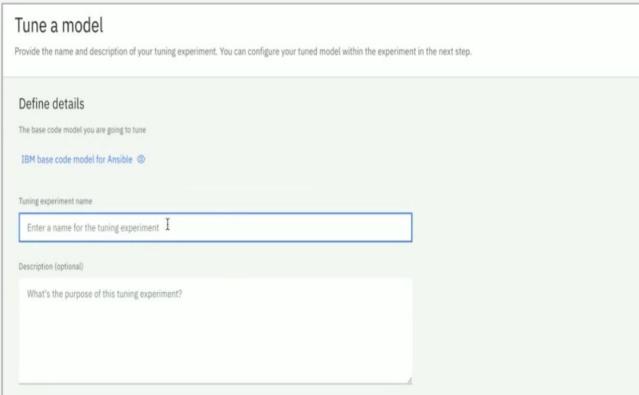
IBM base code model for Ansible ⓘ

Tuning experiment name

Enter a name for the tuning experiment. I

Description (optional)

What's the purpose of this tuning experiment?



## Create

### Model customization / fine tuning

- + Allows Ansible Lightspeed users to leverage their existing Ansible content to train the model.
- + Code recommendations are tailored to the organizations' specific needs and automation patterns.

## Red Hat Ansible Lightspeed

The screenshots illustrate the Ansible Lightspeed interface, showing how it generates playbooks from user input.

**Playbook generation:** A terminal window shows a complex Ansible playbook for installing Docker on Ubuntu 18.04. The playbook includes tasks for adding Docker GPG keys, installing Docker packages, and configuring Docker services. A callout bubble says "Playbook generation".

```
1 # This playbook uses the apt module to install the required packages, the apt_key module to add the Docker GPG key, the
2 module to add the Docker repository, the apt module to install Docker, and the service module to start and enable the Docker service. It then
3 uses the docker_image and docker_container modules to pull the nginx image and run the nginx container, respectively.
4
5 This playbook assumes that:
6   • the target host is running Ubuntu 18.04 and has Docker installed.
7   • the user has the necessary permissions to install and run Docker containers.
8
9
10
11 --- Explain playbook
12 - name: Install Docker
13 hosts: all
14 becomes: true
15 tasks: true
16
17 - name: Install required packages Explain task
18   apt:
19     name: "{{item}}"
20     state: present
21   with_items:
22     - apt-transport-https
23     - ca-certificates
24     - curl
25     - software-properties-common
26     - name: Add Docker GPG key
27       apt_key:
28         url: https://download.docker.com/linux/ubuntu/gpg
29         state: present
30
31     - name: Add Docker repository Explain task
32       apt_repository:
33         repo: deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable
34         state: present
35
36     - name: Install Docker
37       apt:
38         name: docker-ce
39         state: present
40
41     - name: Start Docker service Explain task
42       service:
43         name: docker
44         state: started
45         enabled: true
46
47     - name: Run nginx container Explain task
48       hosts: all
49       tasks:
50         - name: Pull nginx image
51           docker_image:
52             name: nginx
53
54     - name: Run nginx container Explain task
55       docker_container:
56         name: nginx
57         image: nginx
58         state: started
59         ports:
60           - "80:80"
```

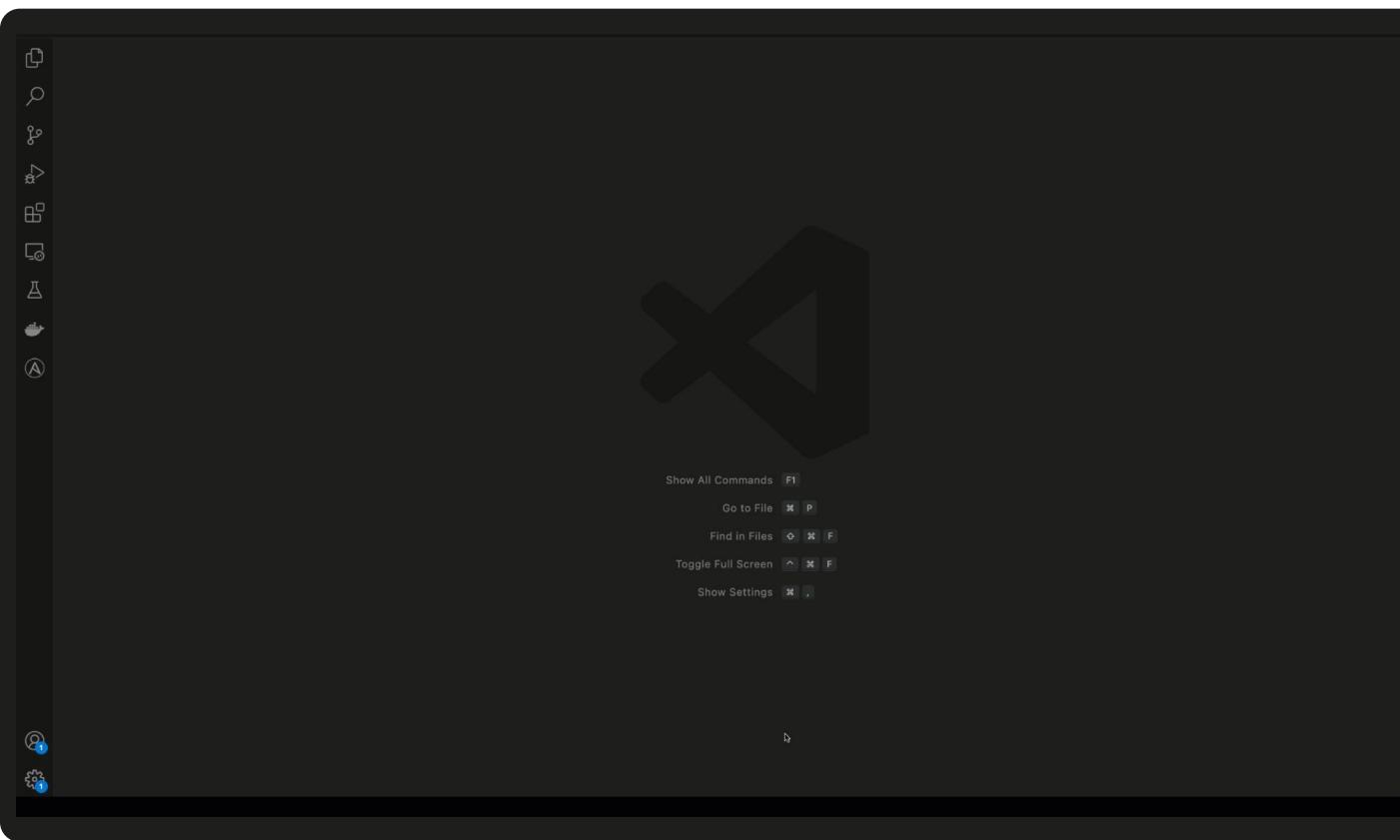
**Guided chat:** A window titled "Create a playbook" asks "What would you want the playbook to accomplish?". A text input field contains the goal: "Install docker, start and enable the service, and run the nginx container on ubuntu". A callout bubble says "Guided chat".

**Prompt refinement:** A window titled "Create a playbook" asks "Does this look like the right playbook for you?". It displays a summary of the steps: "1. Install Docker on all hosts using the apt module. 2. Pull the nginx image from Docker Hub using the docker\_image module. 3. Start a container from the nginx image using the docker\_container module.". A callout bubble says "Prompt refinement".

## Create

### Ansible Playbook generation

- + Guided chat to boost developer productivity
- + Playbook explanation to accelerate learning of syntax and structure
- + WCA 20B Granite model to generate high-quality recommendations



The screenshot shows a dark-themed user interface for a code editor. A large, semi-transparent watermark of the Visual Studio Code logo is centered on the screen. At the top, there's a navigation bar with icons for file operations like Open, Save, Find, and Settings. Below the navigation bar is a menu bar with options like 'File', 'Edit', 'View', 'Insert', 'Select', 'Search', 'Run', 'Terminal', 'Help', and 'About'. A context menu is open at the bottom left, listing options such as 'Show All Commands' (F1), 'Go to File' (P), 'Find in Files' (F), 'Toggle Full Screen' (F), and 'Show Settings' (S). On the right side of the interface, there's a white callout box with a red border containing the word 'Create' in red text. Below 'Create' is a section titled 'Ansible Playbook generation' in bold black text. To the right of this title is a bulleted list of three items, each preceded by a red plus sign (+):

- + Guided chat to boost developer productivity
- + Playbook explanation to accelerate learning of syntax and structure
- + WCA 20B Granite model to generate high-quality recommendations

## Create a role with Ansible Lightspeed

3 of 3

[Learn more about roles](#)

- tasks/main.yml

```
---
- name: Patch rhel servers
  register: rhel_servers_patch_status
  ansible.builtin.package:
    name: "*"
    state: latest

- name: Check to see if a reboot is needed
  register: needs_restarting
  changed_when: needs_restarting.rc == 1
  failed_when: needs_restarting.rc not in (0
  ansible.builtin.command: dnf needs-restart
- name: Reboot rhel servers if needed
  when: needs_restarting.rc == 1
  ansible.builtin.reboot:
```

- defaults/main.yml

### Create a r

2 of 3

[Learn more about roles](#)

"Patch all RHEL servers using DNF and check to see if they need a reboot and reboot if necessary" [Edit](#)  
 Role name:

Review the suggested steps for your role and modify as needed.

1. Patch rhel servers
2. Check to see if a reboot is needed
3. Reboot rhel servers if needed

[Continue](#) [Back](#)

Role  
generation

## Create a role with Ansible Lightspeed

1 of 3

[Learn more about roles](#)

Guided  
chat

Describe what you want to achieve in natural language

Patch all RHEL servers using DNF and check to see if they need a reboot and reboot if necessary

Analyze

### Examples

Create IIS websites on port 8080 and 8081 and open firewall

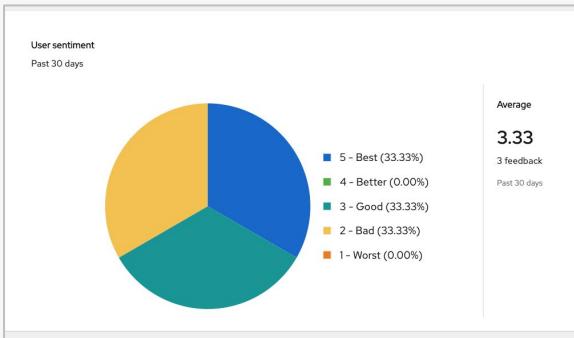
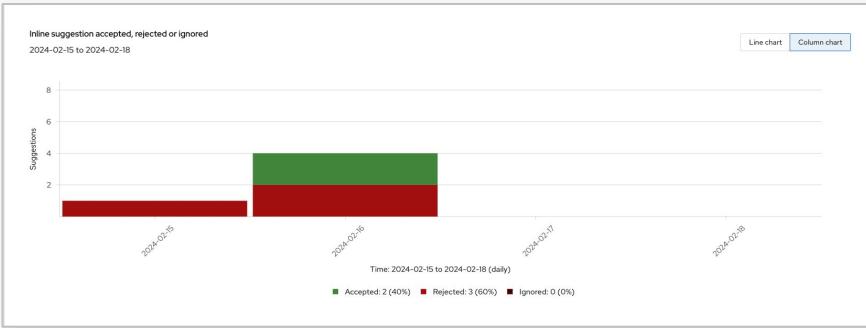
Create a security group named web-servers in AWS, allowing inbound SSH access on port

Create

## Ansible Role generation

- + Guided chat to boost developer productivity
- + Role explanation to accelerate learning of syntax and structure
- + WCA 20B Granite model to generate high-quality recommendations

Prompt  
refinement



Top 10 modules returned in code recommendations  
2024-02-15 to 2024-02-18

Module name | Returned | %

A table titled 'Top 10 modules returned in code recommendations' for the period from February 15 to 18, 2024. It lists the module name, the number of returns, and the percentage of total modules used. The data shows that 'ansible.builtin.package' is the most frequently recommended module.

Module name	Returned	%
ansible.builtin.package	99	(9.26%)
ansible.posix.authorized_key	67	(0.04%)
ansible.builtin.set_fact	59	(1.48%)
ansible.builtin.template	45	(8.75%)
ansible.builtin.command	29	(5.64%)
ansible.builtin.copy	24	(4.87%)
ansible.builtin.service	23	(4.47%)
ansible.posix.fwuid	18	(3.50%)
amazon.aws.ec2.vpc_subnet_info	18	(3.50%)
ansible.builtin.user	14	(2.72%)

Top 10 out of total modules used 77%

## Maintain

### Administrative dashboard

- + Monitoring metrics and usage reporting
- + Understand usage patterns across teams
- + Gain insights to ensure successful adoption
- + Accessed in the Red Hat Hybrid Cloud Console

The screenshot shows a GitHub repository page for the 'network.bgp' role. At the top, there are navigation links: Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the header, the repository name 'network.bgp' is shown as public, forked from 'amugagadu/network\_bgp'. The main branch is 'main' with 1 commit, 1 branch, and 0 tags. The commit history lists 134 commits by 'anshulbehl' over the last week, with the most recent commit being 'Update ansible-code-bot.yml'. Other commits include 'Update actions to operations', 'Prepare Release 1.1.0', and 'Update dire structure'. The repository has a 'GPL-3.0 license' and 0 stars. It also includes sections for 'About', 'Releases' (no releases published), 'Packages' (no packages published), and 'Languages' (Python 100%).

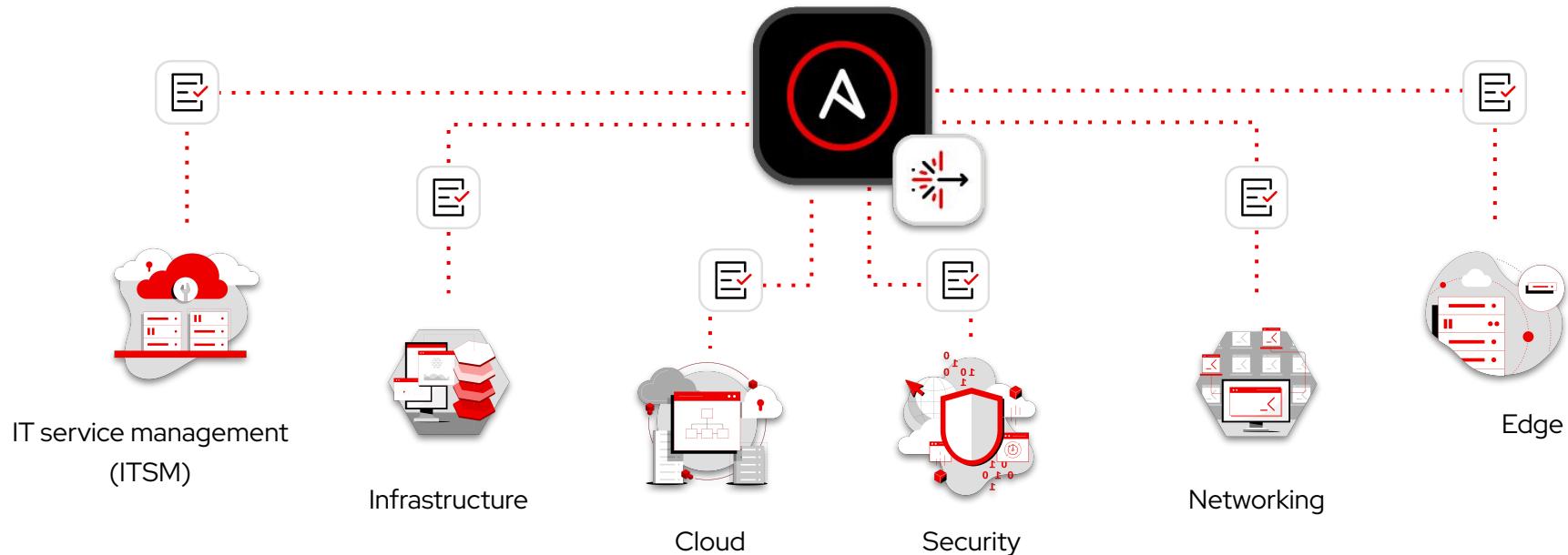
## Maintain

### Ansible code bot

- + Integrates natively with user-selected Git repos, scans existing Ansible content, and automatically submits pull requests with code improvement recommendations
- + Makes it easier for teams to review, test, and apply improvements to the automation code base

Accelerate automation content development across domains

More efficient, consistent creation



# Benefits for the entire IT automation team

Regardless of skill level

AUTOMATION

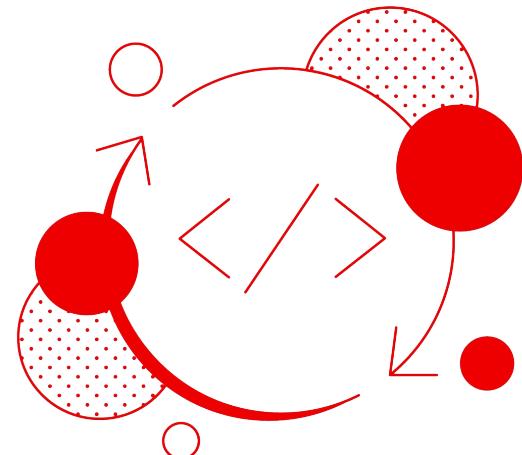
## Novices

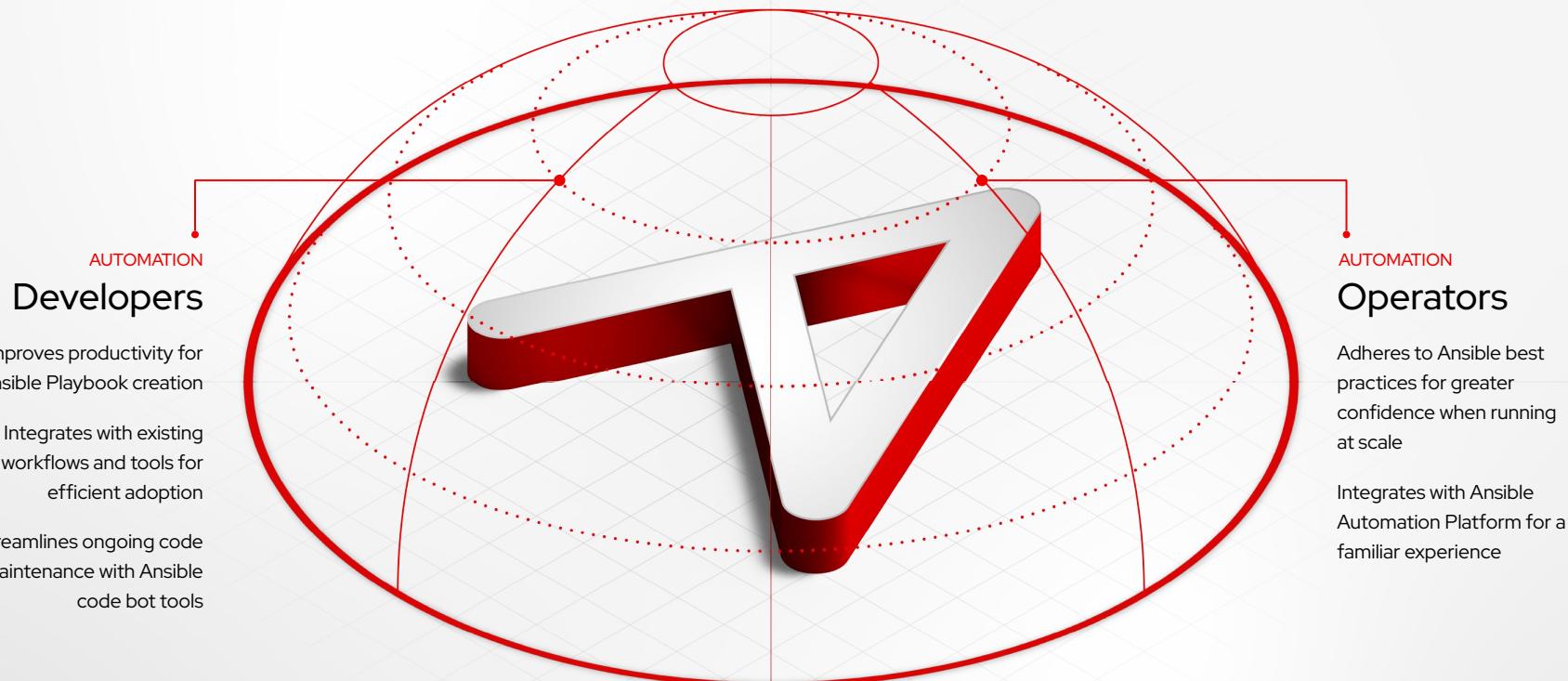
- + Reduces the Ansible and YAML learning curve
- + Helps ensure that the automation content created follows best practices
- + Increases skill development and confidence

AUTOMATION

## Subject matter experts

- + Bridges the gap between automation ideas and Ansible content creation
- + Uses a data model trained across automation domains, extending expertise





# Using generated AI reduced playbook development effort by over 50% in IBM CIO Organization Pilot

IBM watsonx Code Assistant for Red Hat Ansible Lightspeed technical preview generated over half of the playbook content



60%

Of IBM's Ansible Playbook content was automatically generated by watsonx Code Assistant for Red Hat Ansible Lightspeed in tech preview

10X

Expected increase in members of IBM CIO team who can help produce Ansible Playbooks going forward

“Our team of people who could produce Ansible Playbooks before the technical preview program began was around 10. Once the Enterprise Version of watsonx Code Assistant for Red Hat Ansible Lightspeed becomes Generally Available,

Bob Epstein  
Leader  
Content  
we can ask our entire team of 100 to product Ansible content.”

IBM CIO Hybrid Cloud Platforms

# Transforming the way developers learn and work

How IBM Consulting achieved a 30% reduction  
in Ansible Playbook development effort while  
maintaining quality, compliance, and resiliency



Up to  
**45%**

Improvement in initial build  
productivity for Ansible Playbooks

# Decrease

In time it takes a new user to get up to  
speed and be productive in Ansible

“ During the technical preview for watsonx Code Assistant for Red Hat Ansible Lightspeed, we observed initial build productivity improvements in the range of 20%-45%. As we move to GA, even more gains in productivity are expected. We also believe there’s additional downstream productivity that has not been fully quantified yet. Not only are we aiming to accelerate the development phase for Ansible automations and shorten time-to-value for our clients after GA, but we intend for the quality of the content to be higher.”

**Gerry Leitão**

Partner and Global Hybrid Cloud Automation  
IBM Consulting

# Ansible Lightspeed service offering roadmap

## Now

### Model customization / tuning

Use watsonx Code Assistant to create custom Ansible recommendation models

### Full playbook generation

Chat-based interactive interface that allows for better automation generation

### On-premise deployment

For local and air-gapped environments

### Playbook explanation

Understand the impact of your automation code.

### Admin dashboard

Reporting and usage metrics

### Simplified activation of WCA service

90-day trial

## Near term

### Integrations with Red Hat AI portfolio

Enable architecture(s) that utilize open-sourced models supported by Openshift AI and RHEL AI .

### Add'l Ansible code bot Git repo support

GitLab

### Ansible code bot pull request explanations

Understand the impact of suggested repo changes.

## Long term

### Chat bot for VS Code extension

Full-featured chat interface embedded into the coding experience

### AAP chat bot

Integration of chat interface into Ansible automation controller

### Content discovery

Find existing Ansible content instead of writing from scratch

### Rulebook recommendations

Support recommendations for Event-Driven Ansible

### Add'l Ansible code bot Git repo support

Azure DevOps, Bitbucket, etc.





## Ansible Lightspeed resources

Ansible Lightspeed homepage ▶

Getting started video ▶

Getting started blog ▶

Datasheet ▶

Self-paced interactive lab ▶





## Mark your calendar. Make your mark.

We're headed to Boston, Massachusetts, for Red Hat® Summit and AnsibleFest. Join us **May 19-22, 2025**, to connect with community, build new skills, and make your mark on open source.

Register at [red.ht/summit](https://red.ht/summit)



**Enter code: Automate25**  
during registration to unlock  
a \$200 discount per pass!





**Red Hat**

Ansible Automation  
Platform

# Thank you

Red Hat is the world's leading provider of enterprise open source software solutions.

Award-winning support, training, and consulting services make

Red Hat a trusted adviser to the Fortune 500.



[linkedin.com/company/red-hat](https://linkedin.com/company/red-hat)



[youtube.com/user/RedHatVideos](https://youtube.com/user/RedHatVideos)



[facebook.com/redhatinc](https://facebook.com/redhatinc)



[twitter.com/RedHat](https://twitter.com/RedHat)