



# IT Automation

## CI/CD for IaC

### [Practice]

※In this Document “Exastro IT Automation” will be written as “ITA”.

# Table of contents

## 1. Introduction

- 1.1 [About this document](#)
- 1.2 [Environment](#)
- 1.3 [Scenario](#)
- 1.4 [Preparation](#)

## 2. Scenario

- 2.1 [Register Remote repository](#)
- 2.2 [Register Registered account](#)
- 2.3 [Register file link](#)
- 2.4 [Dry run \(No.1\)](#)
- 2.5 [Edit Playbook](#)
- 2.6 [Dry run \(No.2\)](#)
- 2.7 [Execute to Target server](#)

# 1. Introduction



# 1.1 About this document

## Main Menu

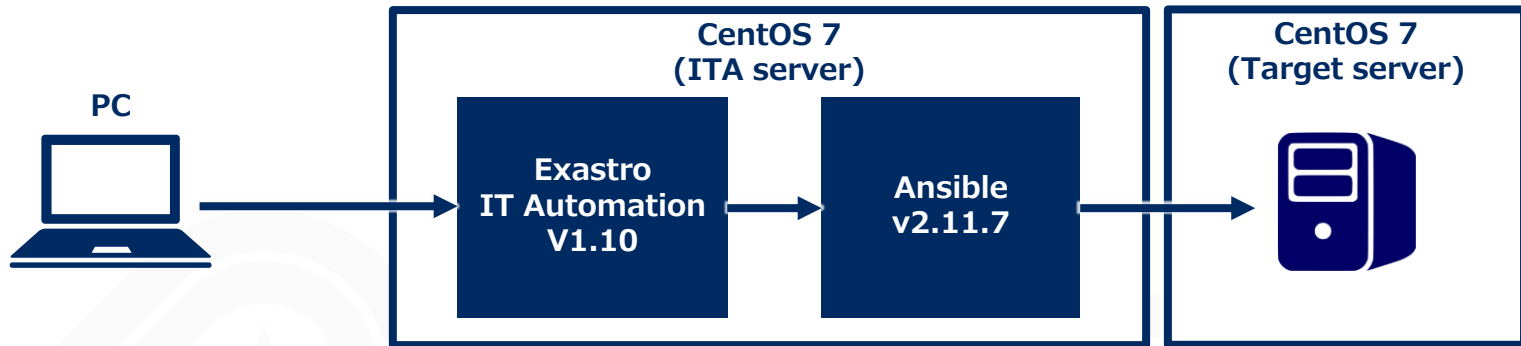
- This document aims to teach the reader about the CI/CD for IaC function by guiding them through a simple scenario.
- This document requires the reader to have finished the scenario in the "[Quickstart](#)" guide before they can follow this scenario.



# 1.2 Environment

## Operation environment

- The environment used in this document is as follows.



### ITA server

- CentOS 7 (※1)
- Exastro ITA v1.10
- Ansible 2.11.7

### Target server

- CentOS 7

※1 This guide uses CentOS7 for the Host server, but ITA can run on any RHEL7 and RHEL8 type OS.

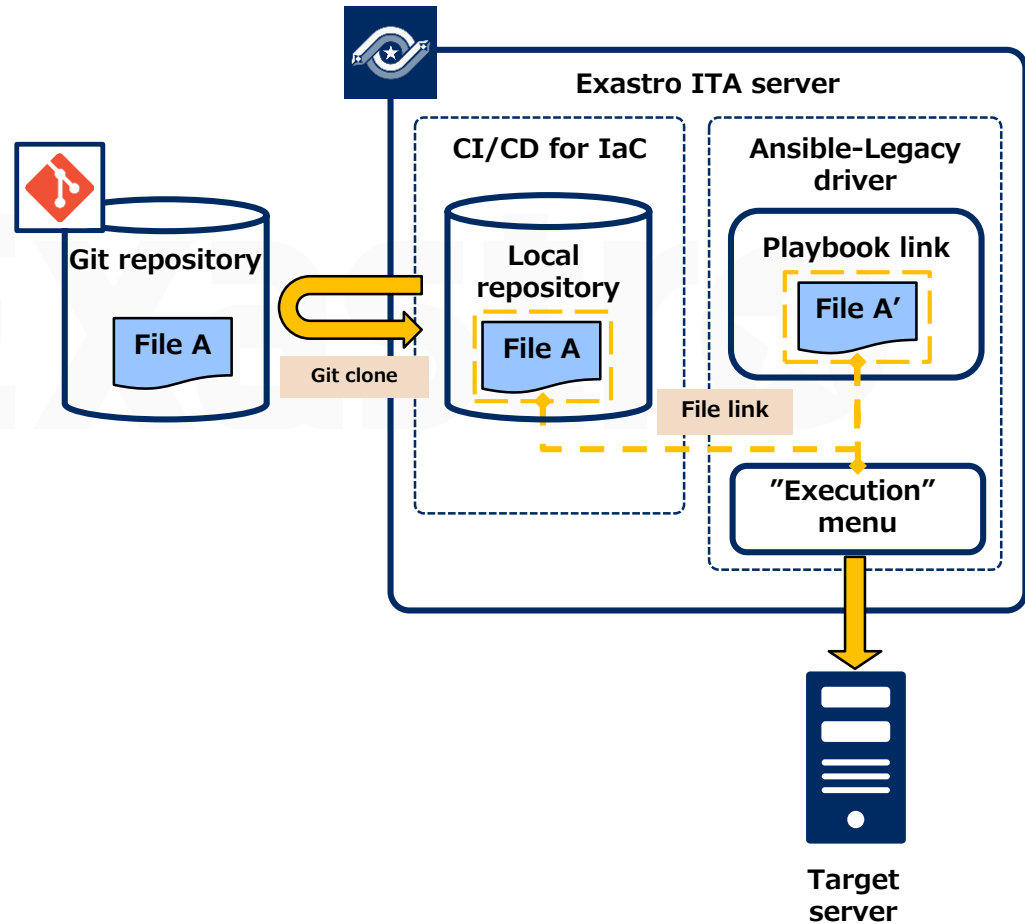
- The user will also need a GitHub account in addition to the environment shown above.

# 1.3 Scenario

## Scenario

- The figure below illustrates the steps and contents of this document's scenario.

- ① Register remote repository
- ② Register registration account
- ③ Register file link
- ④ Dry run(No.1)
- ⑤ Edit Playbook file
- ⑥ Dry run (No.2)
- ⑦ Execute to target server

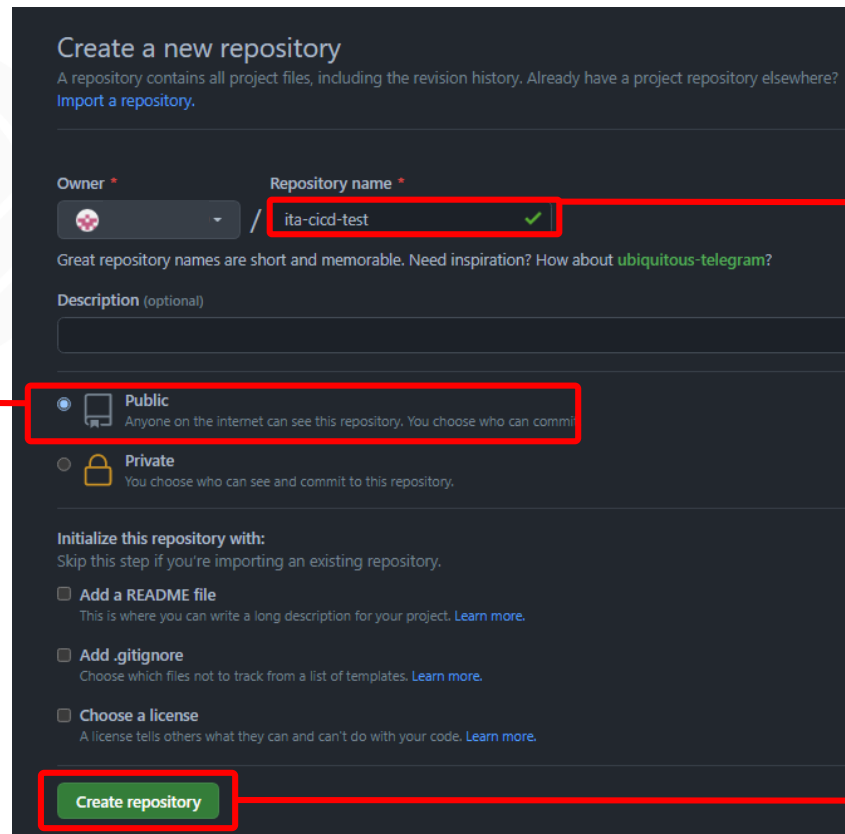


# 1.4 Preparation(1/3)

## Prepare Git repository

- In this scenario, we will use GitHub.

Go to Repository and select “New” to create a new repository.



The screenshot shows the GitHub 'Create a new repository' form. The form is titled 'Create a new repository' and includes a sub-header 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.' Below this, there are fields for 'Owner' (a dropdown menu) and 'Repository name' (a text input field containing 'ita-cicd-test' with a green checkmark). A red box highlights the 'Repository name' field, with a red line pointing to a callout box on the right that says 'Input a name'. Below the name field is a description field with the text 'Description (optional)'. There are two radio button options for visibility: 'Public' (selected) and 'Private'. A red box highlights the 'Public' option, with a red line pointing to a callout box on the left that says 'Select "Public"'. Below the visibility options is a section titled 'Initialize this repository with:' with three checkboxes: 'Add a README file', 'Add .gitignore', and 'Choose a license'. At the bottom of the form is a green 'Create repository' button, which is highlighted with a red box and a red line pointing to a callout box on the right that says 'Click "Create repository"'. The background of the slide features a faint watermark of a graduation cap and a star.

Input a name

Select "Public"

Click  
"Create repository"

# 1.4 Preparation(2/3)

## Prepare Playbook file

- In this scenario, we will use the following Playbook

yum\_package\_install\_check.yml

```
- name: install the latest version of packages
  yum:
    name: "{{ item }}"
    state: latest
  with_items:
    - "{{ VAR_packages }}"

- name: Check yum list
  shell:yum list installed | grep "{{ item }}"
  register: result
  with_items:
    - "{{ VAR_packages }}"
```

### Point

※The file has an invalid indent here on purpose.  
We will fix it later as a part of the scenario.

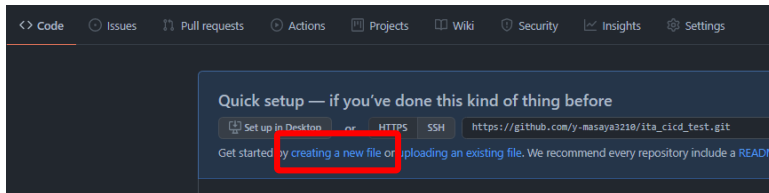


# 1.4 Preparation(3/3)

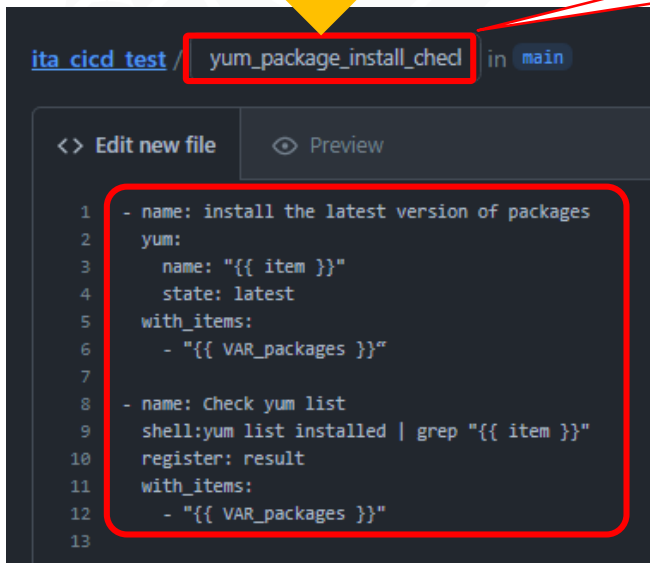
## Upload Playbook

● Upload the Playbook to GitHub.

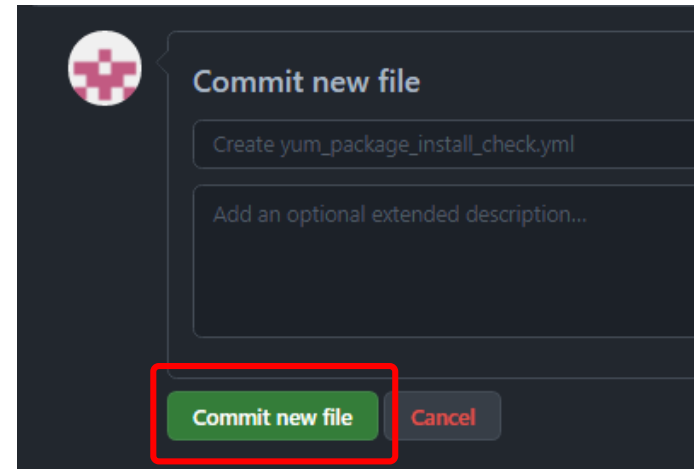
- ① Press the "Code" tab and click "Creating a new file"
- ② Edit a new file and input the contents and name from [the previous slide](#).
- ③ Press "Commit new file".



**Repository name**  
yum\_package\_install\_check.yml  
See this page for the contents.



Scroll down  
→



## 2. Scenario

# 2.1 Register Remote repository

## Register the Git repository information.

- In this step, we will register the information of the GitHub account we prepared earlier. In ITA, go to the "CI/CD for IaC" menu and click "Remote repository". Follow the table below and register a new item.

The screenshot shows the Exastro IT Automation interface. The top navigation bar includes the Exastro logo, 'CI/CD for IaC', and user information: 'User name [System Administrator]', 'Login ID [administrator]', and buttons for 'Role', 'Change password', and 'Logout'. A left sidebar menu has 'Remote repository' highlighted. The main content area shows a 'Register' form with a table for repository details. The table has columns for Item, Repository Name, Remote Repository (URL), Branch, Protocol, Visibility type, Git Account information (User, Password), Last update date/time, and Last updated by. A 'Register' button is highlighted with a red box. A red arrow points from the 'Register' button to the summary table below.

Remote Repository name	Remote Repository (URL)	Protocol	Visibility type	Remote repository sync info(Sync cycle)
(Name of the repository name)	(URL of the repository name)	https	public	Valid

## 2.2 Register Registered account

### Register the account information needed to access the cloned files.

- Register a new item with the Exastro ITA account information.

In this scenario, we will use the “administrator” user

Press “Registered account” and follow the table below to create a new item.

Menu

- Main menu
- Interface information
- Remote repository
- Remote repository file
- Registered account**
- File link

Item number	Exastro IT Automation account		Access permission		Remarks	Last update date/time	Last updated by
	Login Id*	Login Password*	Setting	Role to allow access			
Auto-input	1:administrator	••••••••	Setting			Auto-input	Auto-input

※ \* is a required item.

Back Register

Login ID	Login PW
administrator	(Password)

## 2.3 Register file link(1/2)

### Register a file link for the source files and the cloned files.

- In this section, we will link the source files and the cloned files and register an Operation and a Movement that will check the validity of the cloned files. Go to the "file link" menu and create a new item using the table below.

Menu

- Main menu
- Interface information
- Remote repository
- Remote repository file
- Registered account
- File link**

Description ▾Open

Display filter ▾Open

List/Update ▾Open

Register ▲Close

Git Repository (From)

Item number	Link file name*	Remote Repository*	File path*	Link file type	Last update date/time	Last updated by
Auto-input	yum_package_instal	ita-cicd-test	yum_package_install_check.yml	Ansible-Legacy console/Playbook files	Auto-input	Auto-input

※\*is a required item.

Back Register

Link destination file name	Remote repository	File path	Link destination file type	Execution login ID	Automatic synchronization
yum_package_instal	ita_cicd_test	yum_package_instal_check.yml	Ansible-Legacy console /Playbook files	1:administrator	Valid

## 2.3 Register file link(2/2)

### Register Operation and Movement and select "Dry run".

- After having filled out the items from the previous slide, scroll to the right and fill out the following 3 items and press "Register".

Menu

- Main menu
- Interface information
- Remote repository
- Remote repository file
- Registered account
- File link**

Item number	Synchronization	Operation	Movement	Dry run	Access permission	Last update date/time	Last updated by
1		OP2	Package install	<input type="radio"/>	Setting	Auto-input	Auto-input

※\* is a required item.

Back Update

Operation	Movement	Dry run
Operation 2	Package install	<input checked="" type="radio"/>

## 2.4 Dry run (No.1)(1/2)

### Check that the operation has dry run.

- Linking files will automatically start a dry run.

Go to the Ansible Legacy menu and click “Execution list”. From there, click “Filter” to see all executed operations. Find the operation we dry ran earlier and press the “Operation status check” to see the error contents.

The screenshot shows the Ansible Legacy interface. On the left is a navigation menu with the following items: Menu, Main menu, Movement list, Playbook files, Movement playbook link, Substitution value auto-registration setting, Target host, Substitution value list, Execution, and Check operation status. The 'Execution list' item is highlighted with a red box.

The main content area is titled 'Description' and contains a 'Display filter' section. This section includes a table with columns: Discard, Execution No., Execution type, Status, execution engine, Last update date/time, and Last updated by. Below the table are 'Filter' and 'Clear filter' buttons, and a checked 'Auto-filter' checkbox.

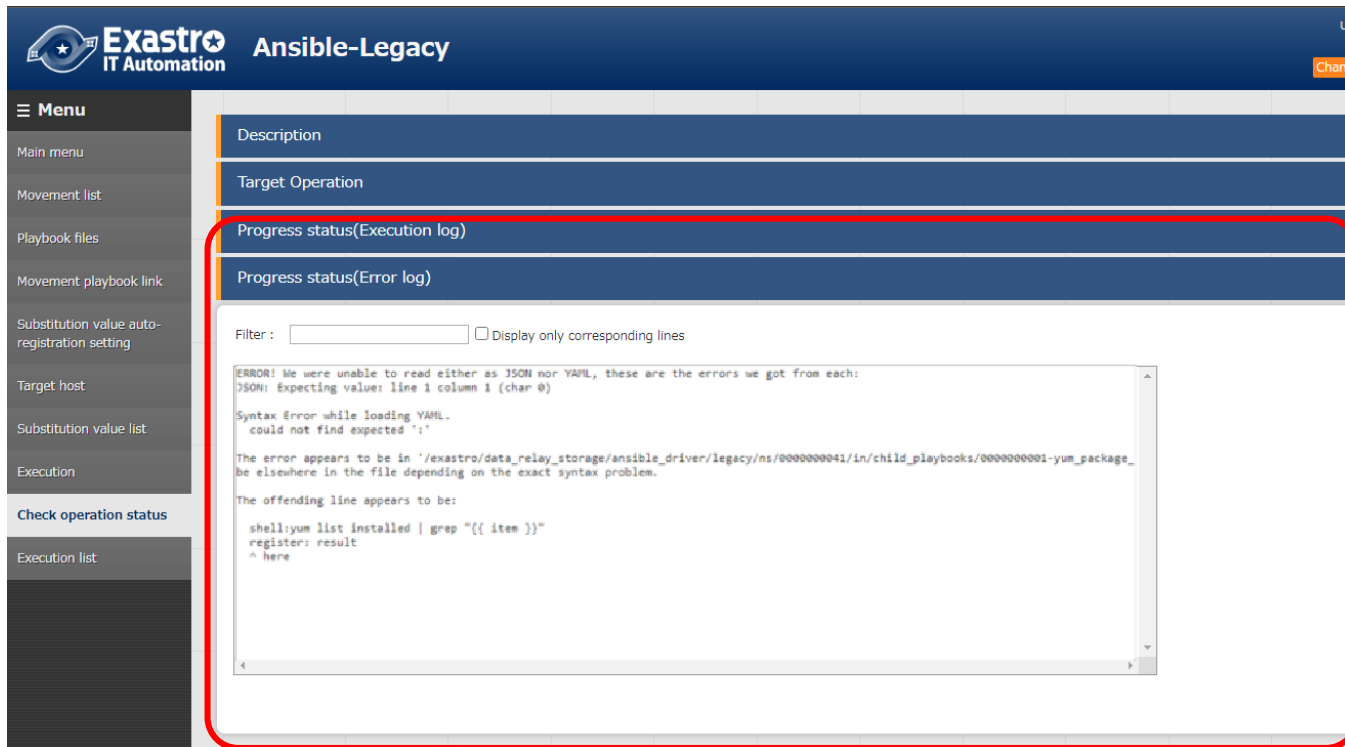
Below the filter section is a 'List' section containing a table of execution records. The table has columns: History, Execution No., Check execution status, Execution type, Status, execution engine, virtualenv, Caller, Last update date/time, and Last updated by. The row for Execution No. 12 is highlighted with a red box, showing a status of 'Unexpected error'.

History	Execution No.	Check execution status	Execution type	Status	execution engine	virtualenv	Caller	Last update date/time	Last updated by
History	12	Check execution status	Normal	Unexpected error	Ansible Engine		InstallP	2021/12/09 13:36:39	Collection work procedure
History	11	Check execution status	Normal	Completed	Ansible Engine			2021/10/18 14:33:01	Collection work procedure
History	10	Check execution status	Normal	Completed	Ansible Engine			2021/09/02 13:13:57	Collection work procedure
History	9	Check execution status	Normal	Completed (error)	Ansible Engine			2021/09/02 13:11:56	Collection work procedure
History	8	Check execution status	Normal	Unexpected error	Ansible Engine			2021/09/02 13:09:45	Collection work procedure

## 2.4 Dry run (No.1)(2/2)

### Check that the operation has dry run.

- Scroll down to see the Progress log. Users can use this to see the contents of the error. As mentioned earlier, the file contained an invalid indent which caused the error.



The screenshot displays the Exastro IT Automation Ansible-Legacy web interface. The left sidebar contains a 'Menu' with options like 'Main menu', 'Movement list', 'Playbook files', 'Movement playbook link', 'Substitution value auto-registration setting', 'Target host', 'Substitution value list', 'Execution', 'Check operation status', and 'Execution list'. The main content area shows a table with columns for 'Description', 'Target Operation', 'Progress status(Execution log)', and 'Progress status(Error log)'. The 'Progress status(Error log)' section is highlighted with a red border and contains a text area with the following error message:

```
ERROR! We were unable to read either as JSON nor YAML, these are the errors we got from each:
JSON: Expecting value: line 1 column 1 (char 0)

Syntax Error while loading YAML.
could not find expected ':'

The error appears to be in '/exastro/data_relay_storage/ansible_driver/legacy/ns/0000000041/in/child_playbooks/0000000001-yum_package_
be elsewhere in the file depending on the exact syntax problem.

The offending line appears to be:

shell: yum list installed | grep "{{ item }}"
register: result
^ here
```

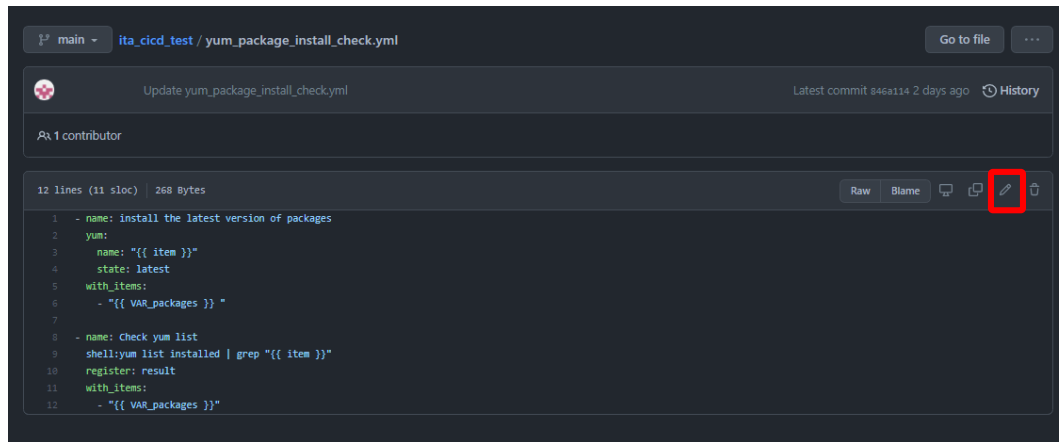


## 2.5 Edit Playbook

### Access GitHub and edit the Playbook file.

- In this section, we will fix the error from the previous slide.

Access the GitHub file and press the edit icon. Follow the steps below and press “Commit changes”.



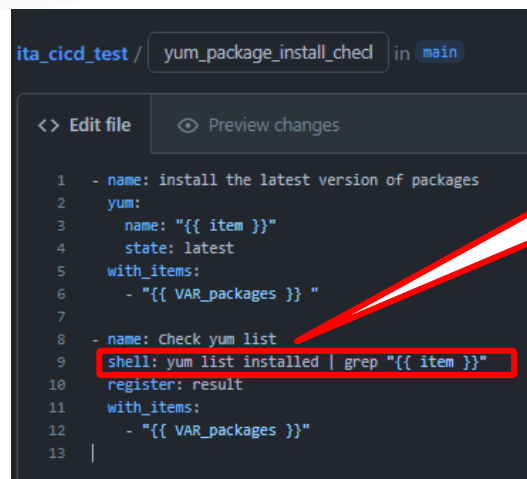
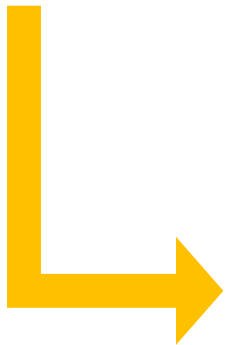
```
1 - name: install the latest version of packages
2 yum:
3   name: "{{ item }}"
4   state: latest
5   with_items:
6     - "{{ VAR_packages }}"
7
8 - name: Check yum list
9   shell: yum list installed | grep "{{ item }}"
10  register: result
11  with_items:
12    - "{{ VAR_packages }}"
```

Input a space between ":" and "y"

```
shell:yum list installed | grep "{{ item }}"
```

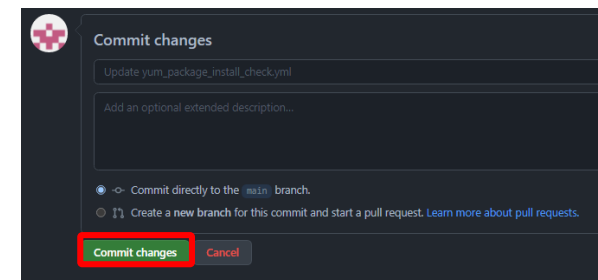


```
shell: yum list installed | grep "{{ item }}"
```



```
1 - name: install the latest version of packages
2 yum:
3   name: "{{ item }}"
4   state: latest
5   with_items:
6     - "{{ VAR_packages }}"
7
8 - name: Check yum list
9   shell: yum list installed | grep "{{ item }}"
10  register: result
11  with_items:
12    - "{{ VAR_packages }}"
13
```

Scroll down



Commit changes

Update yum\_package\_install\_check.yml

Add an optional extended description...

Commit directly to the main branch.

Create a new branch for this commit and start a pull request. Learn more about pull requests.

**Commit changes** Cancel

# 2.6 Dry run (No.2)

## Check that the operation has dry run.

- Updating the GitHub source file will automatically update and dry run the ITA Playbook clone file.

Do the same as Dry run(No.1) and go to "Ansible Legacy"-> "Execution list" and find the operation. The last dry run ended in an error, but this run should end successfully.

The screenshot shows the ITA Ansible Legacy interface. On the left, a sidebar menu has 'Execution list' highlighted with a red box. The main content area is divided into sections: 'Description', 'Display filter', and 'List'. The 'Display filter' section contains a table with columns: Discard, Execution No., Execution type, Status, execution engine, Last update date/time, and Last updated by. Below this table are 'Filter' and 'Clear filter' buttons, with the 'Filter' button highlighted in red. A checkbox for 'Auto-filter' is checked. The 'List' section contains a table with columns: History, Execution No., Check execution status, Execution type, Status, execution engine, virtualenv, Caller, Last update date/time, and Last updated by. The first row of this table is highlighted with a red border.

History	Execution No.	Check execution status	Execution type	Status	execution engine	virtualenv	Caller	Last update date/time	Last updated by
History	2	Check execution status	Normal	Completed	Ansible Engine			2021/09/01 13:28:31	Collection work procedure
History	6	Check execution status	Normal	Completed	Ansible Engine			2021/09/02 09:38:53	Collection work procedure
History	10	Check execution status	Normal	Completed	Ansible Engine			2021/09/02 13:13:57	Collection work procedure

# 2.7 Execute to Target server(1/2)

## Run the operation and apply to the Target server.

- Now that we have used the Dry run to see that there are no problems with the Playbook, we can apply it to the target server. Go to Ansible Legacy > Execution. Here we can select what Movement and Operation that we want to run.

**Menu**

- Main menu
- Conductor interface information
- Conductor class list
- Conductor class edit
- Conductor execution**
- Conductor confirmation
- Conductor list
- Conductor Regularly execution

Description ▽Open

Scheduling △Close

Specify the scheduled date/time in (YYYY/MM/DD HH:MM) Immediately execute when blank.  
Scheduled date/time:

Conductor [filter] ▽Open

Conductor [List] △Close

Select	Conductor class ID	Conductor name	Explanation	Access permission Role to allow access	Remarks	Last update date/time	Last updated by
<input type="radio"/>	1	InstallPackage				2021/12/09 13:09:33	System Administrator

Filter result count: 1

Operation [Filter] ▽Open

Operation [List] △Close

Select	No.	Operation ID	Operation name	Scheduled date for execution	Last execution date	Access permission Role to allow access	Last update date/time	Last updated by
<input type="radio"/>	5	5	OP1	2021/12/08 19:00			2021/12/07 19:01:04	System Administrator
<input type="radio"/>	6	6	OP2	2021/12/09 15:10			2021/12/08 14:48:13	System Administrator
<input type="radio"/>	7	7	Operation 1	2021/12/30 13:32	2021/12/09 13:35		2021/12/09 13:35:38	Legacy execution procedure
<input type="radio"/>	8	8	Operation 2	2021/12/14 14:21			2021/12/09 14:21:36	System Administrator

## 2.7 Execute to Target server(2/2)

### Run the operation and apply to the Target server.

- After selecting the operation and Movement, press "Execute".  
Executing any operation will move the user to a screen where they can see the status of the running operation. If the operation status says "Completed", the operation has ended successfully.

Operation [Filter]

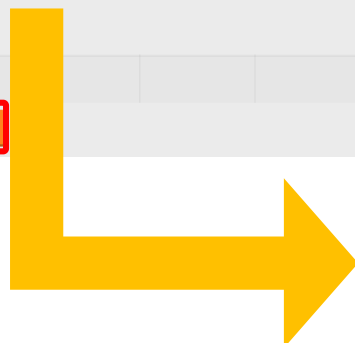
Operation [List]

Select	No.	Operation ID	Operation name	Scheduled date for execution	Last execution date	Access
<input type="radio"/>	5	5	OP1	2021/12/08 19:00		Role to al
<input type="radio"/>	6	6	OP2	2021/12/09 15:10		
<input type="radio"/>	7	7	Operation 1	2021/12/30 13:32	2021/12/09 13:35	
<input checked="" type="radio"/>	8	8	Operation 2	2021/12/14 14:21		

Filter result count: 4

Movement ID 11  
Movement Name Package install

Item	Value	
Execution No.	11	
Execution type	Normal	
Status	Completed	
execution engine	Ansible Engine	
Caller symphony		
Caller conductor		
Execution user	System Administrator	
Movement	ID	11
	Name	<input type="button" value="Package install"/>
	Delay timer (minutes)	
Dedicated information for ansible	Host specific format	IP
	WinRM connection	
Operation	No.	8
	Name	Operation 2
	ID	8
Host management	<input type="button" value="confirmation"/>	
Substitution value	<input type="button" value="confirmation"/>	
Input data	Populated data	<a href="#">InputData 000000011.zip</a>
Output data	Result data	<a href="#">ResultData 000000011.zip</a>
Operation status	Scheduled date/time	
	Start date/time	2021/10/18 14:32:47
	End date/time	2021/10/18 14:32:53





**Exastro**