Automation for Networks and more

Christoff Visser – IIJ Lab
christoff@iij.ad.jp

Internet Initiative Japan

Introduction

- Crash course for commonly used tools for network automation

- Not an in-depth tutorial
  - Give you an idea how you could use automation in your day to day
  - Small version of tutorial hosted by myself and Anurag Bhatia at APNIC 56

- Resources will be available in Gitlab
  - Slides
  - Tutorials
  - Instructions

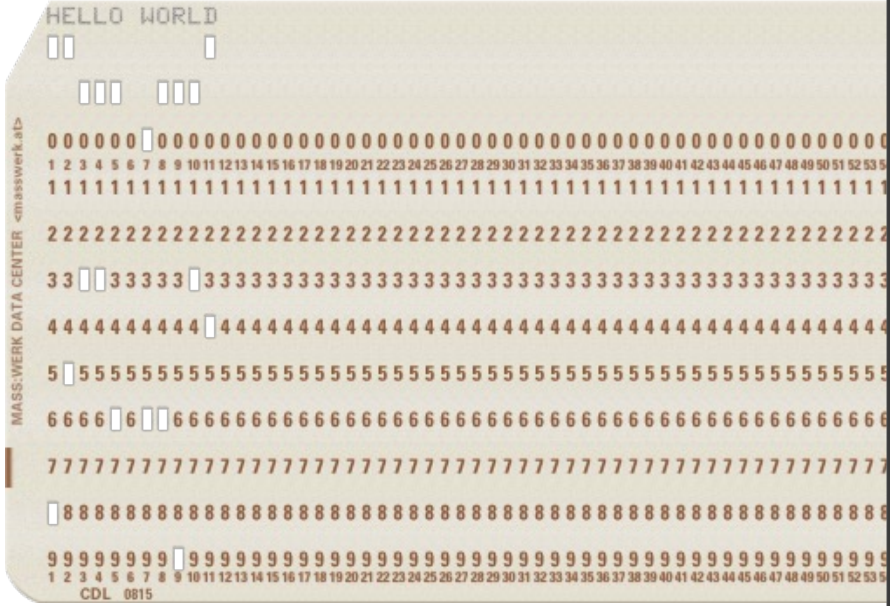
- Link: https://bit.ly/iijlab_automate_techtrend2023

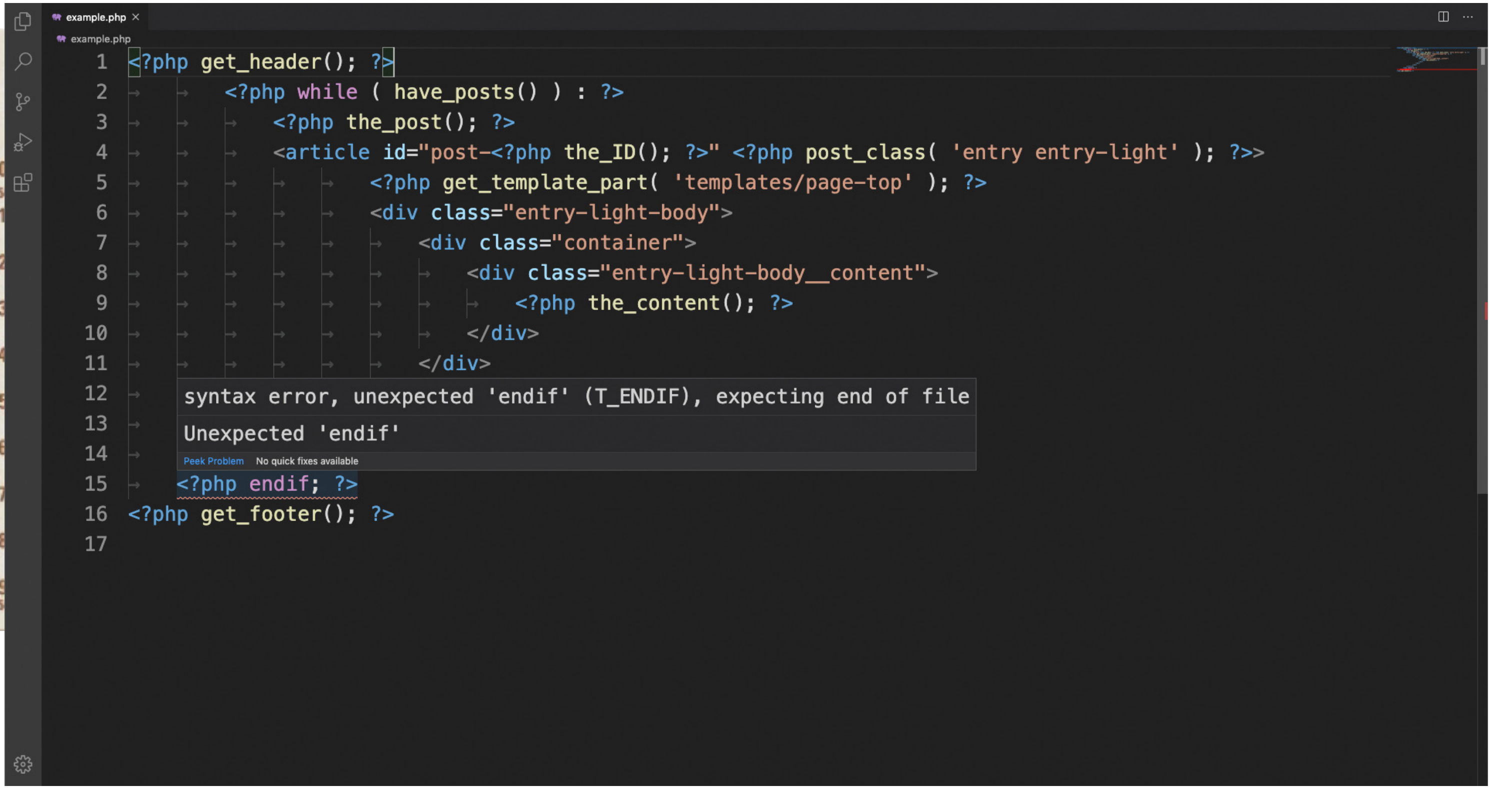
Agenda

- Docker + Containers
- Ansible
- Gitlab + CI/CD
- ChatGPT

Automation as a tool

Automation as a tool

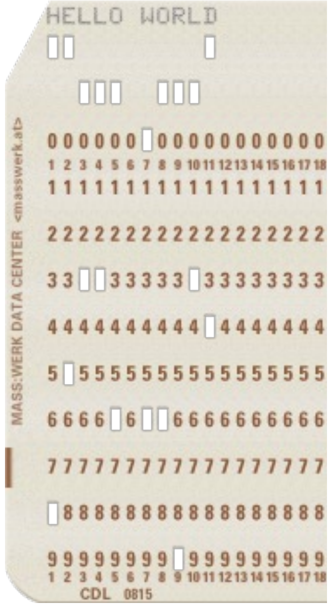




```
<?php get_header(); ?>
        <?php while ( have_posts() ) : ?>
            <?php the_post(); ?>
            <article id="post-<?php the_ID(); ?>" <?php post_class( 'entry entry-light' ); ?>>
                    <?php get_template_part( 'templates/page-top' ); ?>
                    <div class="entry-light-body">
                        <div class="container">
                            <div class="entry-light-body__content">
                                <?php the_content(); ?>
                            </div>
                        </div>
    <?php endif; ?>
<?php get_footer(); ?>
```

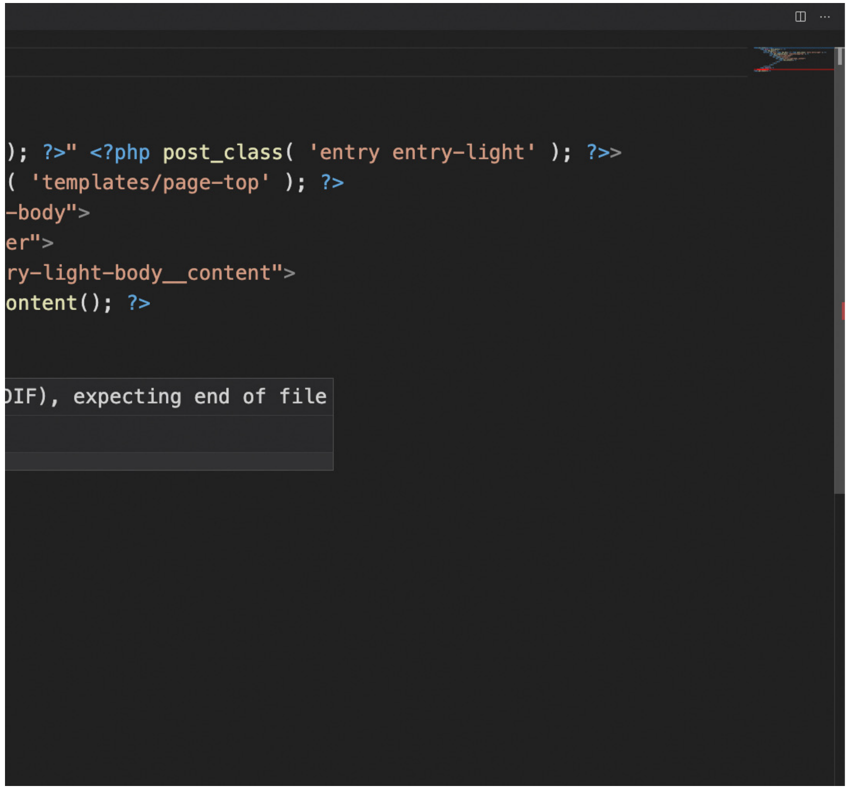
```
syntax error, unexpected 'endif' (T_ENDIF), expecting end of file
Unexpected 'endif'
```

Automation as a tool



Dear autocorrect.

It's never duck.



```
); ?>" <?php post_class( 'entry entry-light' ); ?>>
( 'templates/page-top' ); ?>
-body">
er">
ry-light-body__content">
ontent(); ?>

DIF), expecting end of file
```

Why do we automate?

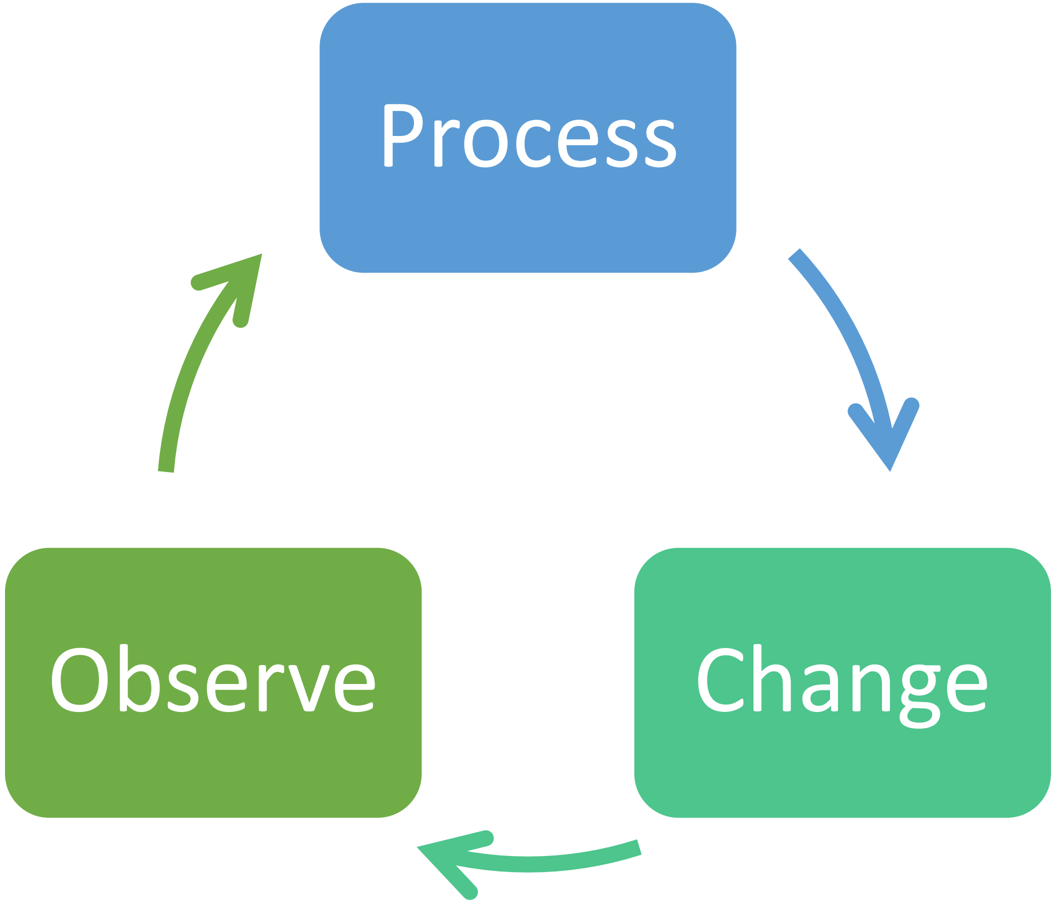
- Increase productivity
- Reduce cost
- Minimise errors
- Increased capability

Why ~~do we~~ should you automate?

- I se productiv
-
- Mini
  ca

Spend less time on boring and tedious tasks

Event driven automation



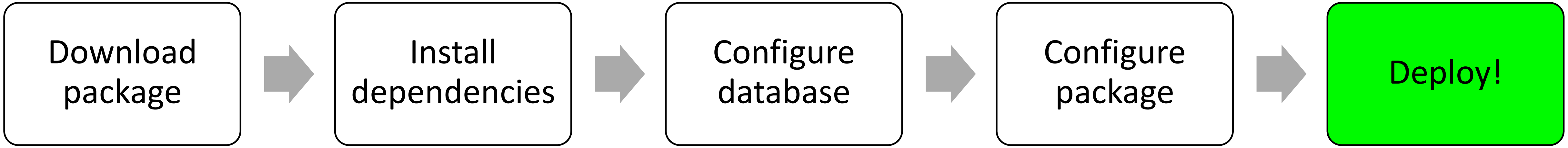
Process

Change

Observe

Docker and containers

Typical software deployment workflow



Download package

Install dependencies

Configure database

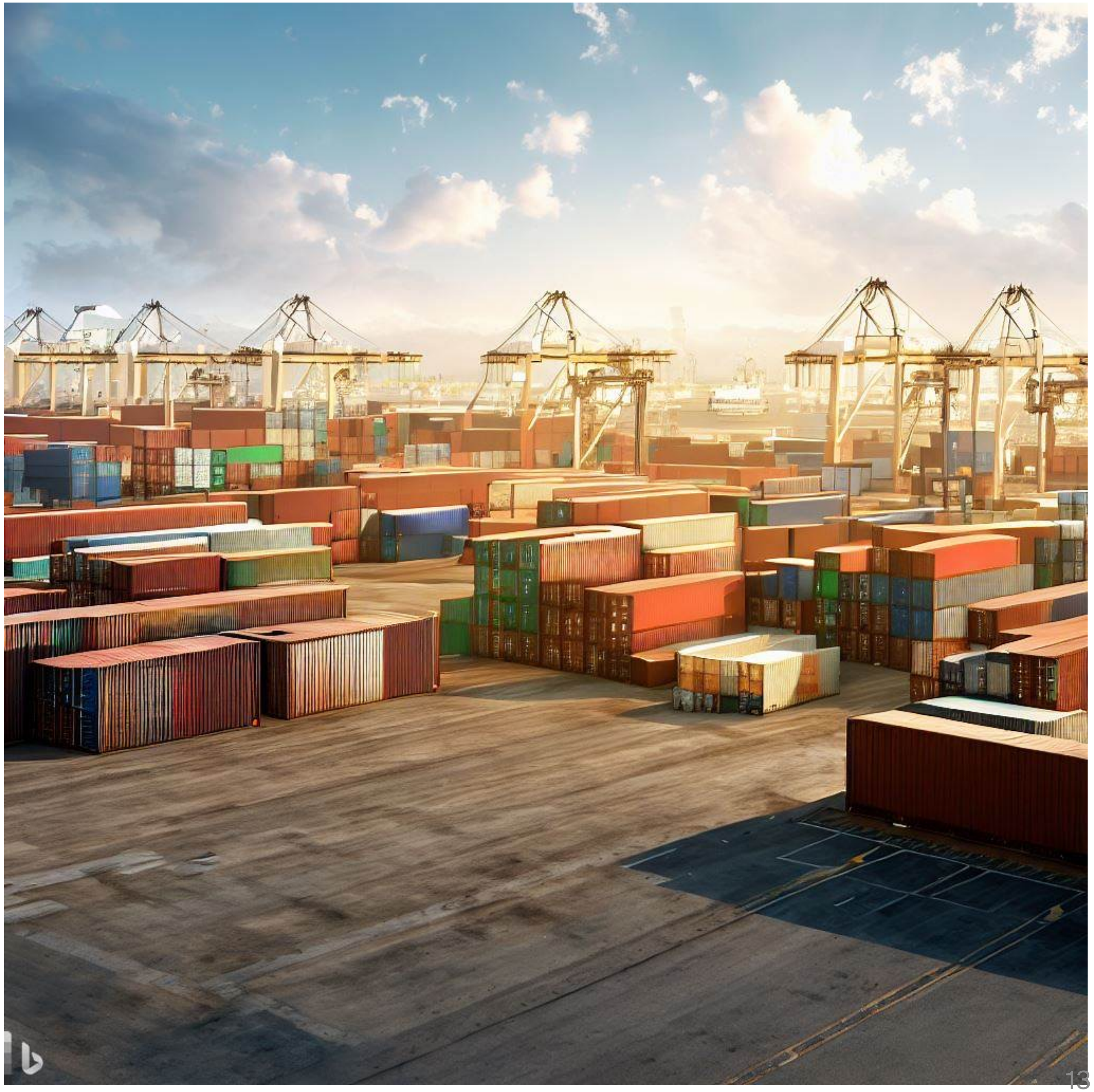
Configure package

Deploy!

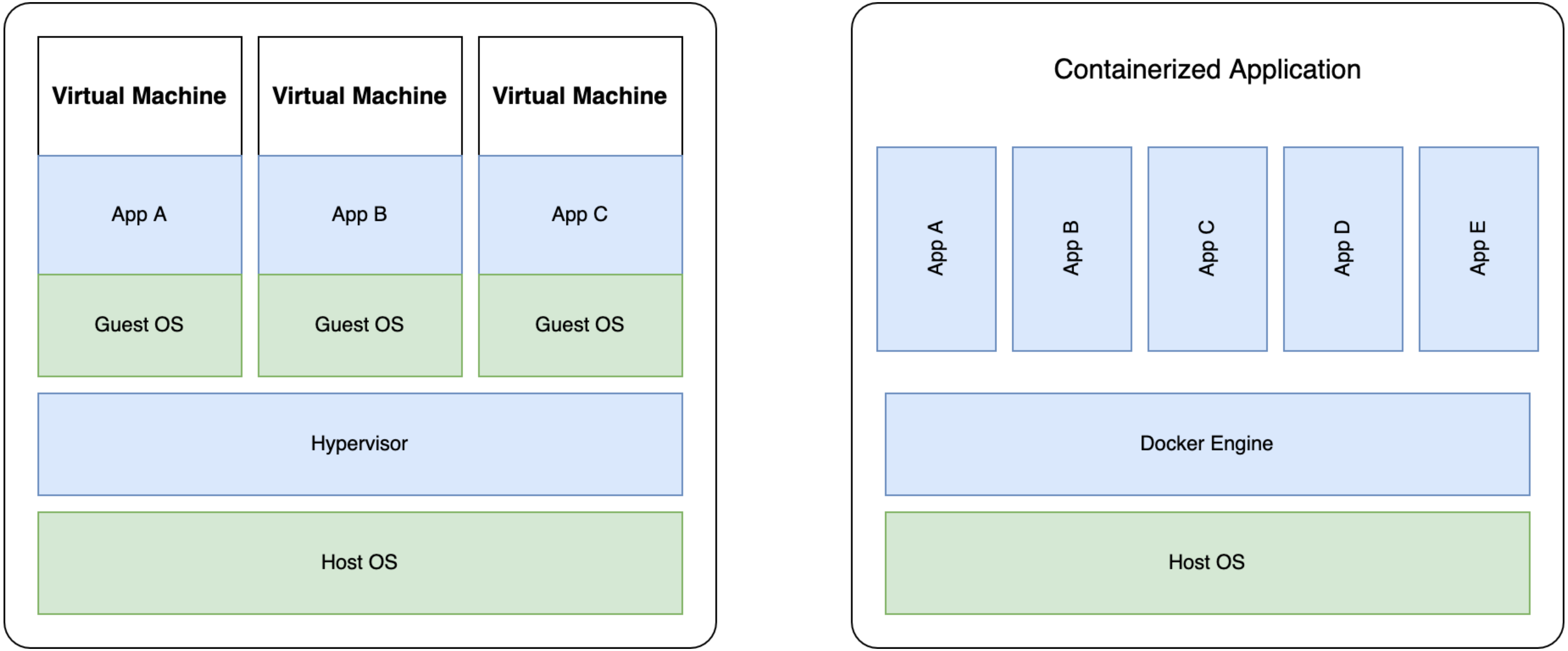
Challenges with traditional software deployment

1. Takes time to go through documentation, install package and maintain it
2. Time consuming process to transfer software to a different server
3. Prone to errors and mistakes
4. “Works on my system”
5. Dependency conflicts

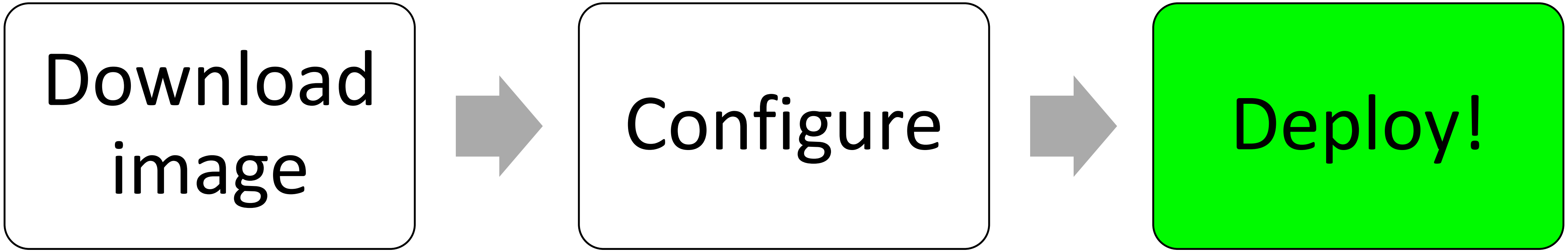
Introducing containers



Virtual Machines vs Containers



Container based software deployment

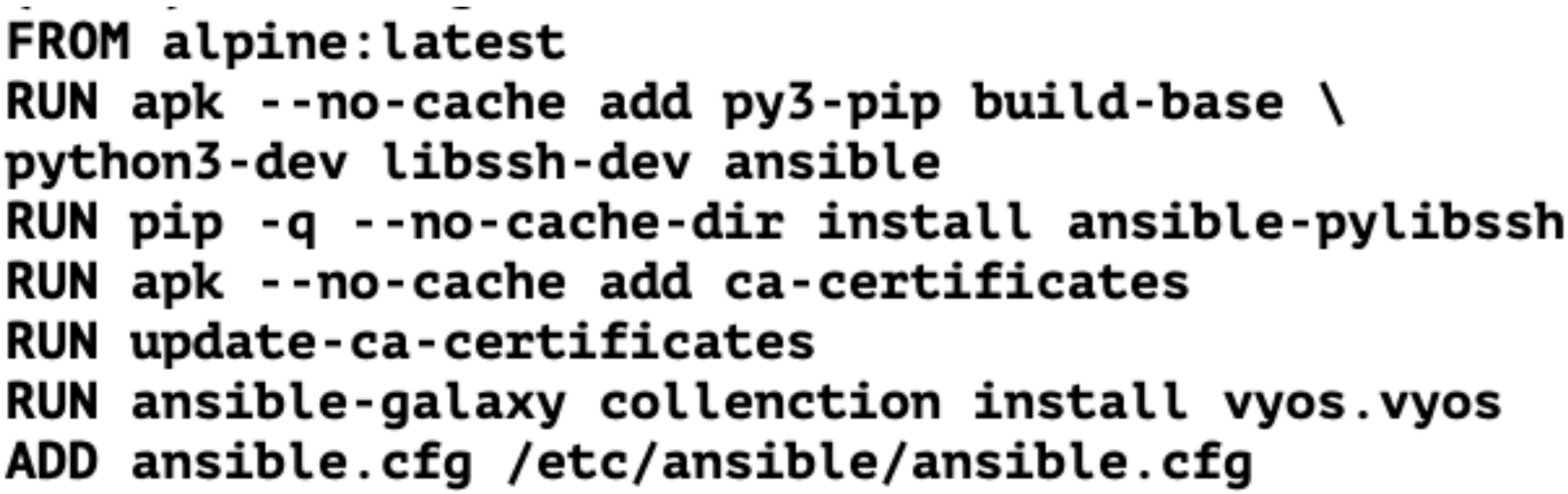


Download image

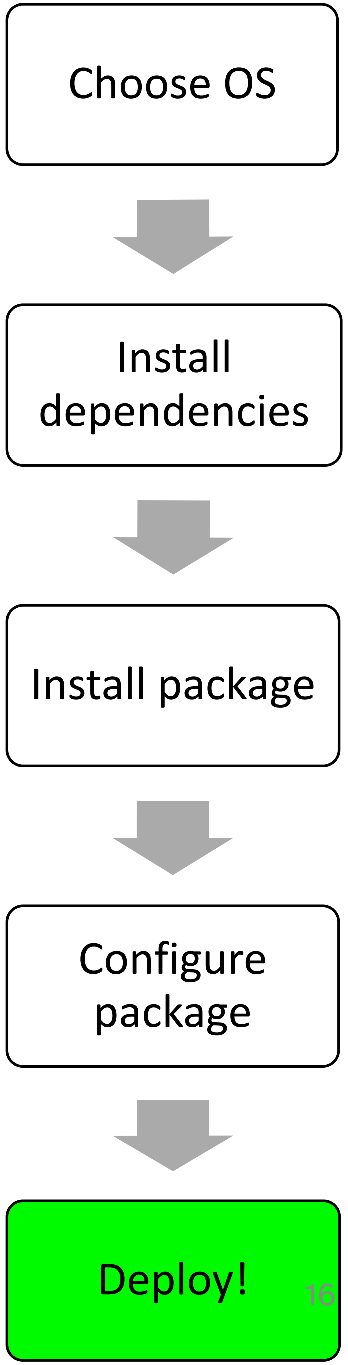
Configure

Deploy!

Building a Container - ansible



```
FROM alpine:latest
RUN apk --no-cache add py3-pip build-base \
python3-dev libssh-dev ansible
RUN pip -q --no-cache-dir install ansible-pylibssh
RUN apk --no-cache add ca-certificates
RUN update-ca-certificates
RUN ansible-galaxy collenction install vyos.vyos
ADD ansible.cfg /etc/ansible/ansible.cfg
```



Choose OS

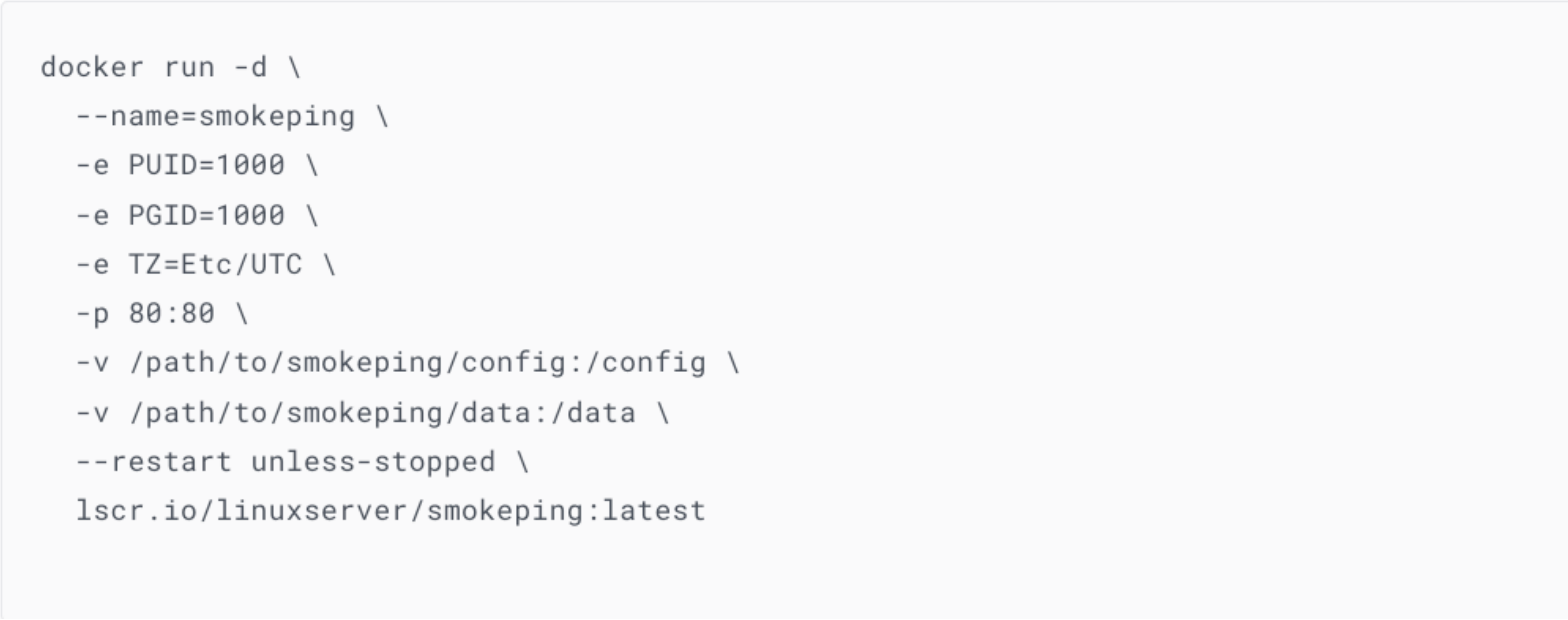
Install dependencies

Install package

Configure package

Deploy!

Running docker ad-hoc



```
docker run -d \
  --name=smokeping \
  -e PUID=1000 \
  -e PGID=1000 \
  -e TZ=Etc/UTC \
  -p 80:80 \
  -v /path/to/smokeping/config:/config \
  -v /path/to/smokeping/data:/data \
  --restart unless-stopped \
  lscr.io/linuxserver/smokeping:latest
```

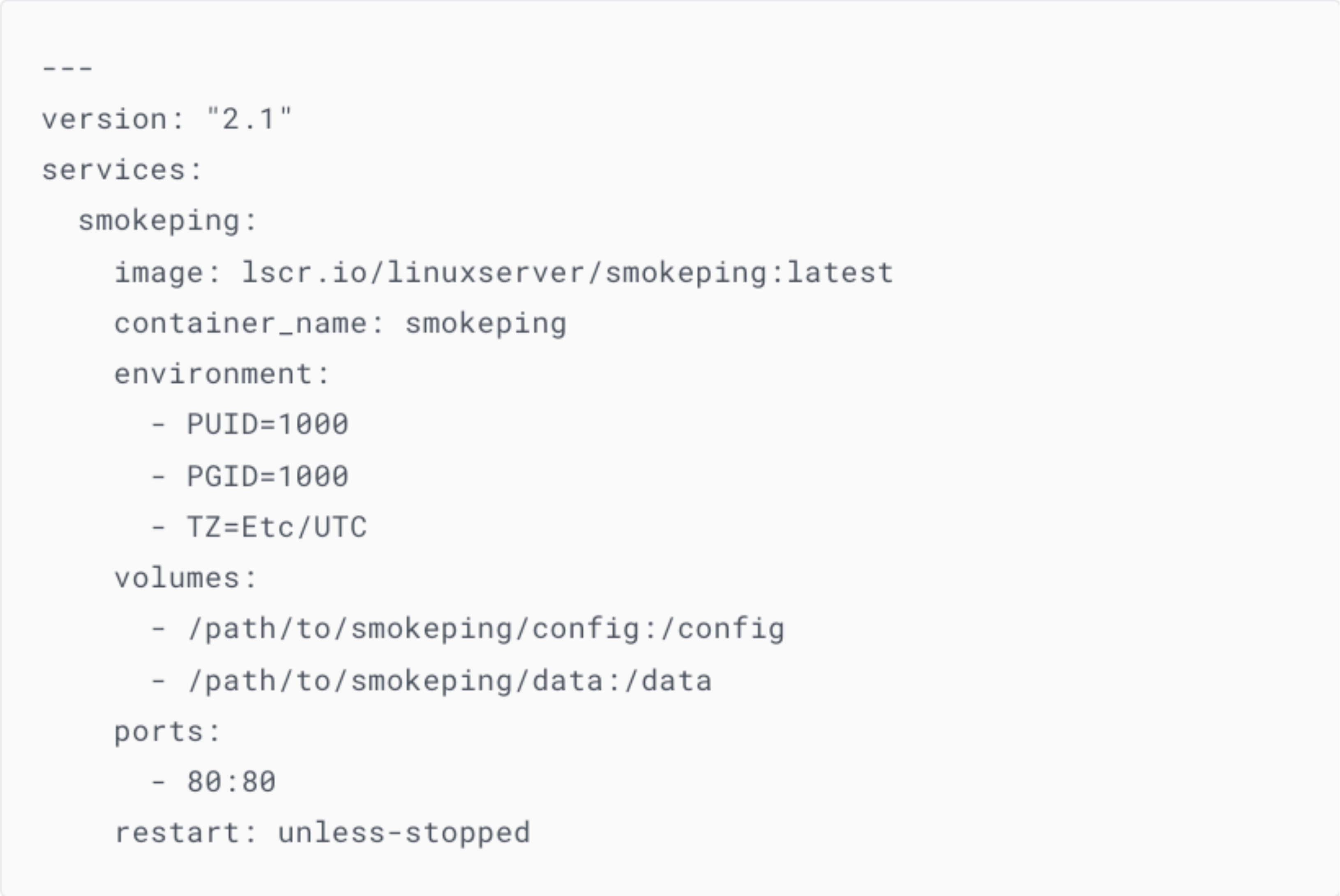
Docker ad-hoc challenges

- Gets complicated the more arguments are passed
- Hard to remember all previously used arguments
- Easy to misconfigure when running multiple ad-hoc containers

docker-compose.yml

- Single yaml text file for multiple containers
- Easier to read and includes all instructions for all containers
- Simplify creating/attaching volumes to bind to
- Ensure you’re exposing only what you need to
- Simplify upgrading and maintaining containers

Docker compose - smokeping



```
---
version: "2.1"
services:
  smokeping:
    image: lscr.io/linuxserver/smokeping:latest
    container_name: smokeping
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Etc/UTC
    volumes:
      - /path/to/smokeping/config:/config
      - /path/to/smokeping/data:/data
    ports:
      - 80:80
    restart: unless-stopped
```

Notes on backup

- Docker containers are reproducible. No need to backup

- User data is stored using volumes or bind mounts
  - Only these need to be backed up

- Popular tools like Restic Duplicati (can be run as docker container)

- Always encrypt data before storing it on the cloud

Some containers to play with

- RIPE Atlas - https://hub.docker.com/r/jamesits/ripe-atlas
- HTML 5 speedtest - https://hub.docker.com/r/adolfintel/speedtest
- iperf3 - https://hub.docker.com/r/networkstatic/iperf3
- Nextcloud - https://hub.docker.com/_/nextcloud
- Docker-speedtest-grafana - https://github.com/frdmn/docker-speedtest-grafana
- Kerberos - https://doc.kerberos.io/opensource/installation#docker
- Nginx Proxy Manager - https://nginxproxymanager.com/
- Linux-server.io - Many great images actively maintained by the open source community

For more information:

- Gitlab repo:
  - More in detailed information
  - Tutorials and labs to follow
  - Guide for minimising the size of docker images

- Link to docker section

ANSIBLE

Automation

- Execute a task without active human interaction
- Commonly done with scripting (bash, python, etc.)
- Focus typically is single task and not overall workflow

Orchestration

- Execute a workflow without human interactions
- Requires a platform to achieve complex workflow executions
- Focus workflow involving multiple tasks, with scheduling, different devices/services etc.
- Combination of automation

Possible tasks to automate/template

- Basic configuration - hostname, SNMP, NTP, AAA, DNS resolvers etc
- Configuration of firewall
- Configuration of BGP session
- Local account management
- Configuration of interface
- Configuration of backup

- And anything else you do regularly

Orchestration tool options

- Ansible
- Saltstack
- Puppet
- Chef
- And more…

ANSIBLE

puppet

CHEF™

Introduction to Ansible

- Popular open source tool for Orchestration
- Agentless i.e. target devices do not need to run any software
- Python based
- Supports various networking devices – Cisco, Juniper, Mikrotik, Huawei, VyOS, Ubnt, Dell and more
- Supports almost every OS – Linux, MacOS, Windows via WSL
- Most are “idempotent”: Only do something when a change is required
- By default comes with CLI & not web UI (Web UI can be added via Ansible AWX or Ansible semaphore)

Ansible Key Concepts

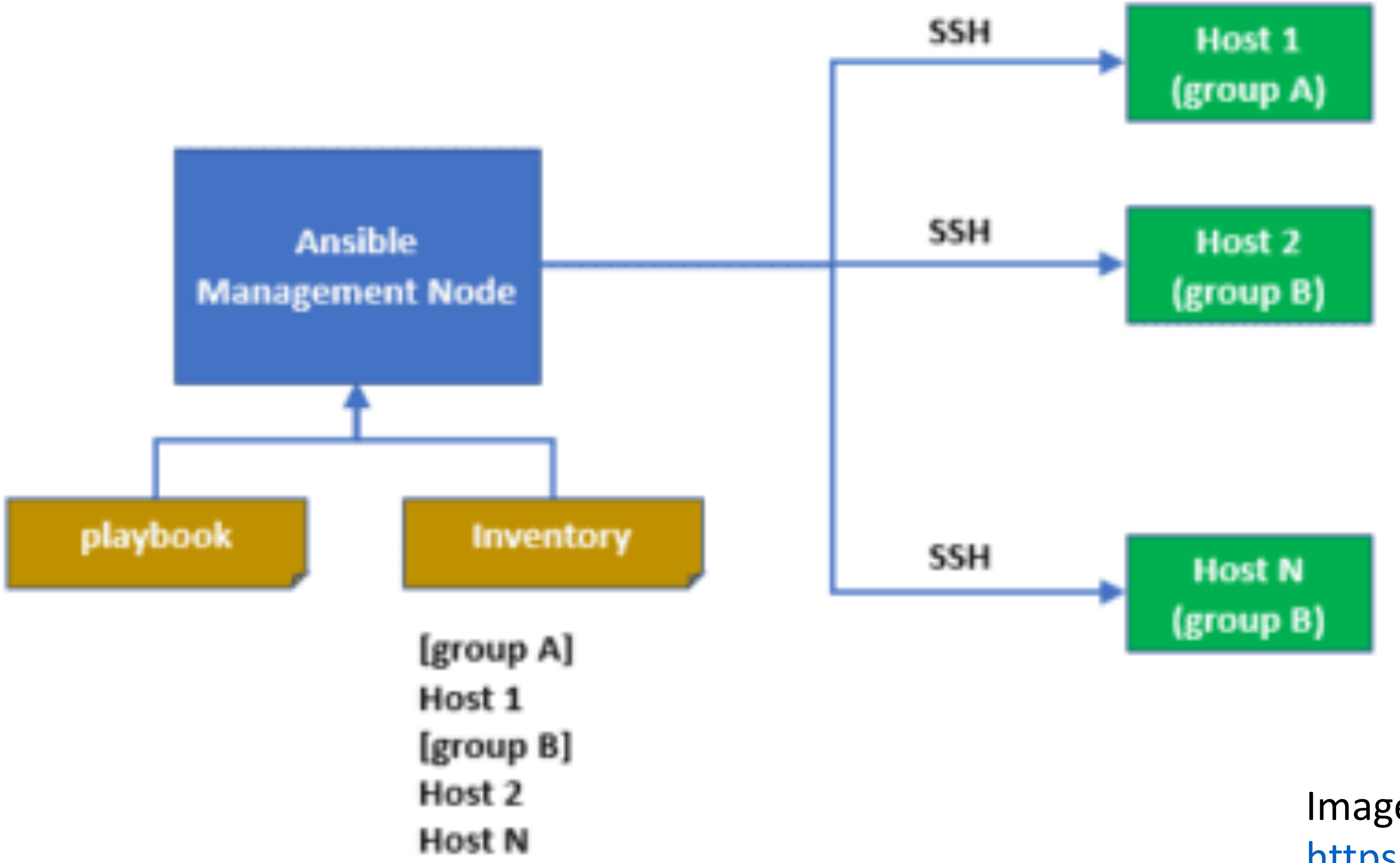


Image source:
https://geekflare.com/ansible-basics/

Writing a playbook

- Written in yaml format (.yml). It’s easy to read format for humans as well as computers

- Yaml can be written in any text editor but there’s risk of breaking yaml syntax and thus a basic code editor like VSCode is a good option

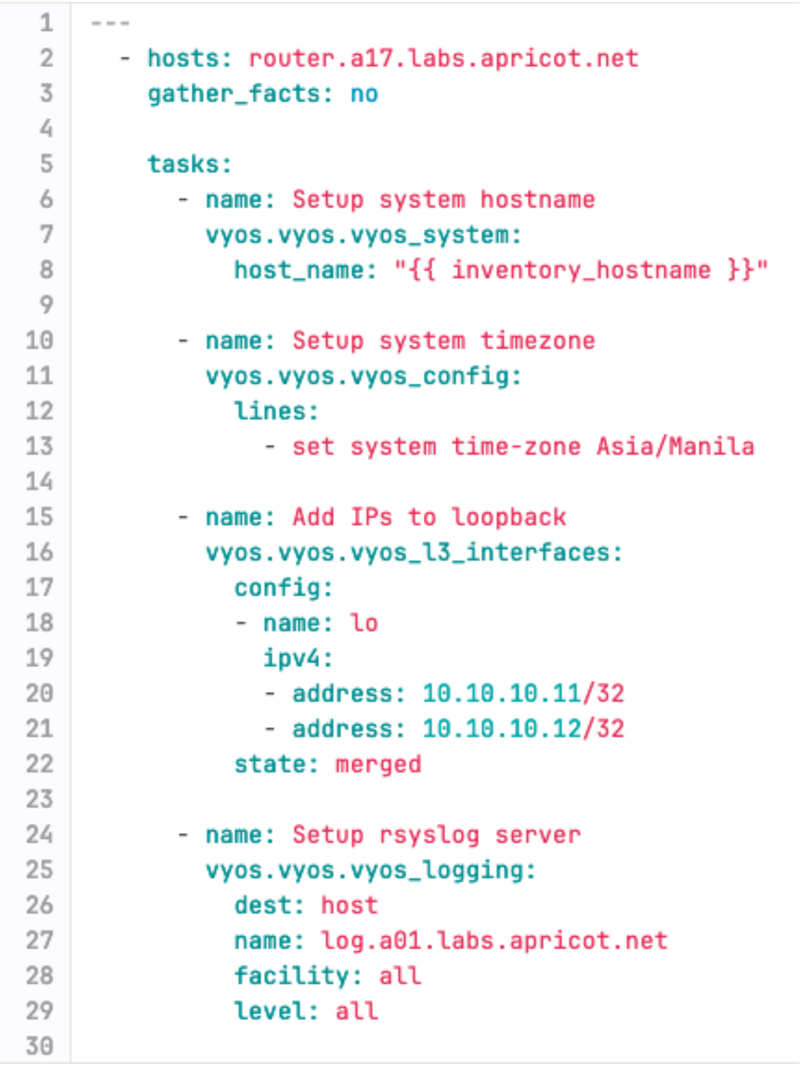
- One can write playbook on locally installed VSCode on Linux/OS X/Windows & save it on remote server using SSH within VSCode

- Use a Ansible extension in VSCode to ensure it throws error if syntax is broken

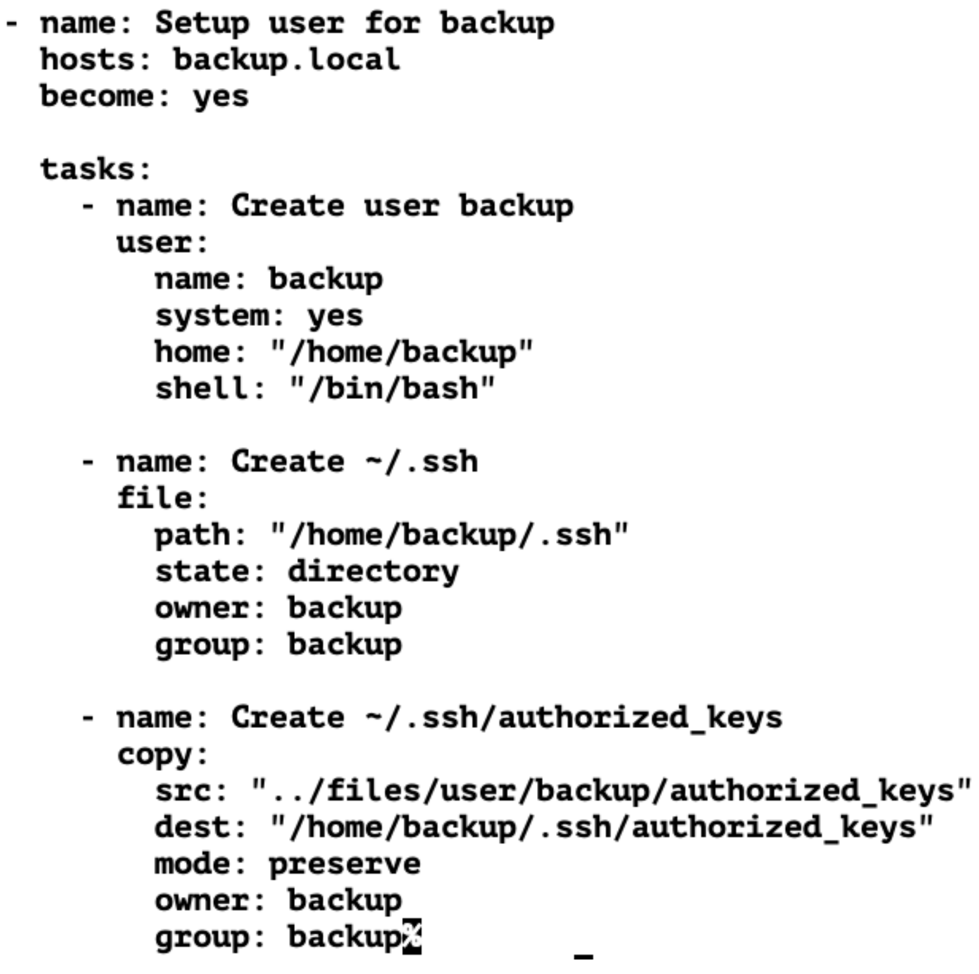
Sample playbook - simple

```
---
 - hosts: routers

   tasks:
    - name: Setup hostname of Router
      vyos.vyos.vyos_system:
        host_name: "{{ inventory_hostname }}"
```

Sample playbook - more complex

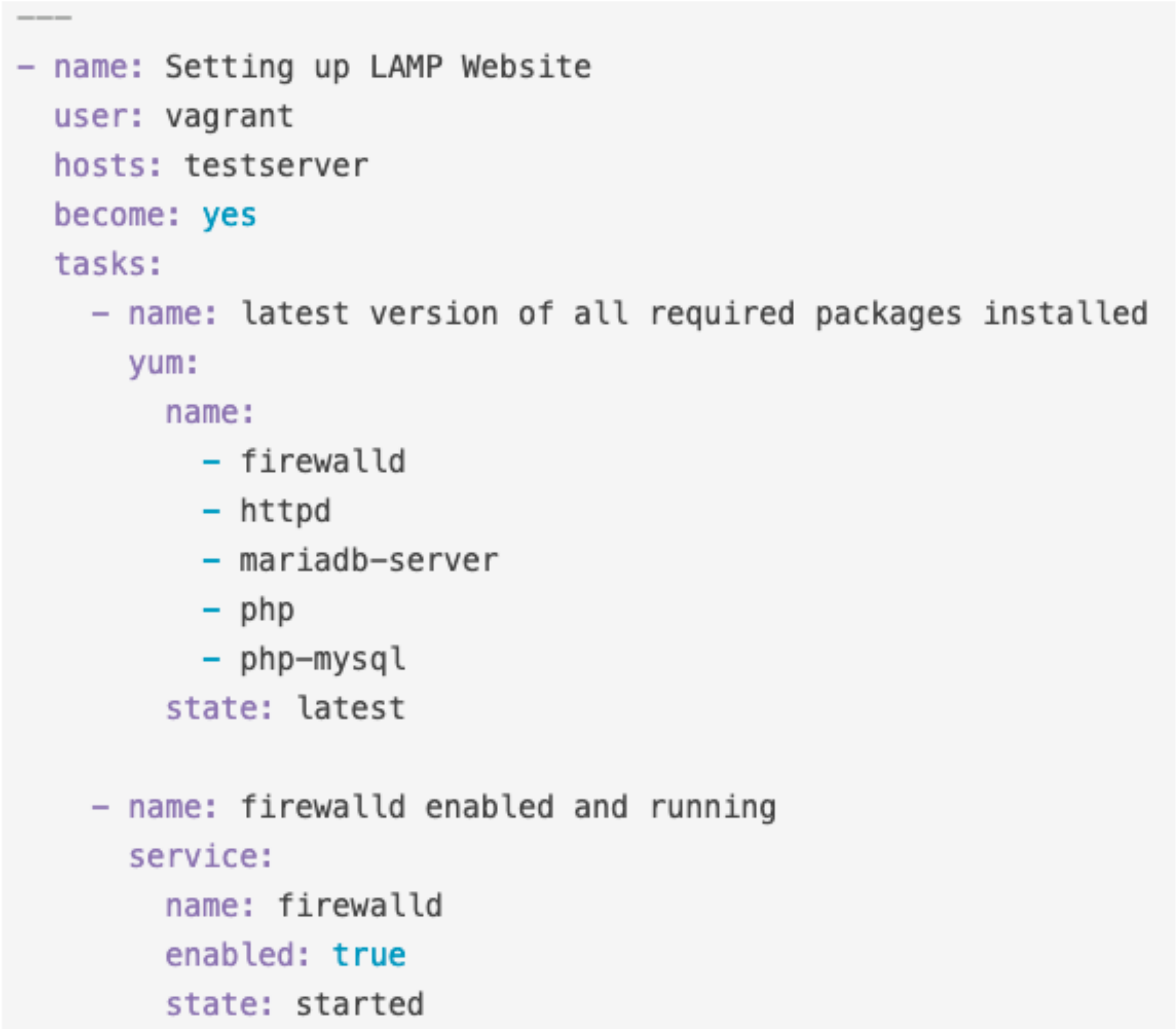


```
---
  - hosts: router.a17.labs.apricot.net
    gather_facts: no

    tasks:
      - name: Setup system hostname
        vyos.vyos.vyos_system:
          host_name: "{{ inventory_hostname }}"

      - name: Setup system timezone
        vyos.vyos.vyos_config:
          lines:
            - set system time-zone Asia/Manila

      - name: Add IPs to loopback
        vyos.vyos.vyos_l3_interfaces:
          config:
          - name: lo
            ipv4:
            - address: 10.10.10.11/32
            - address: 10.10.10.12/32
          state: merged

      - name: Setup rsyslog server
        vyos.vyos.vyos_logging:
          dest: host
          name: log.a01.labs.apricot.net
          facility: all
          level: all

```

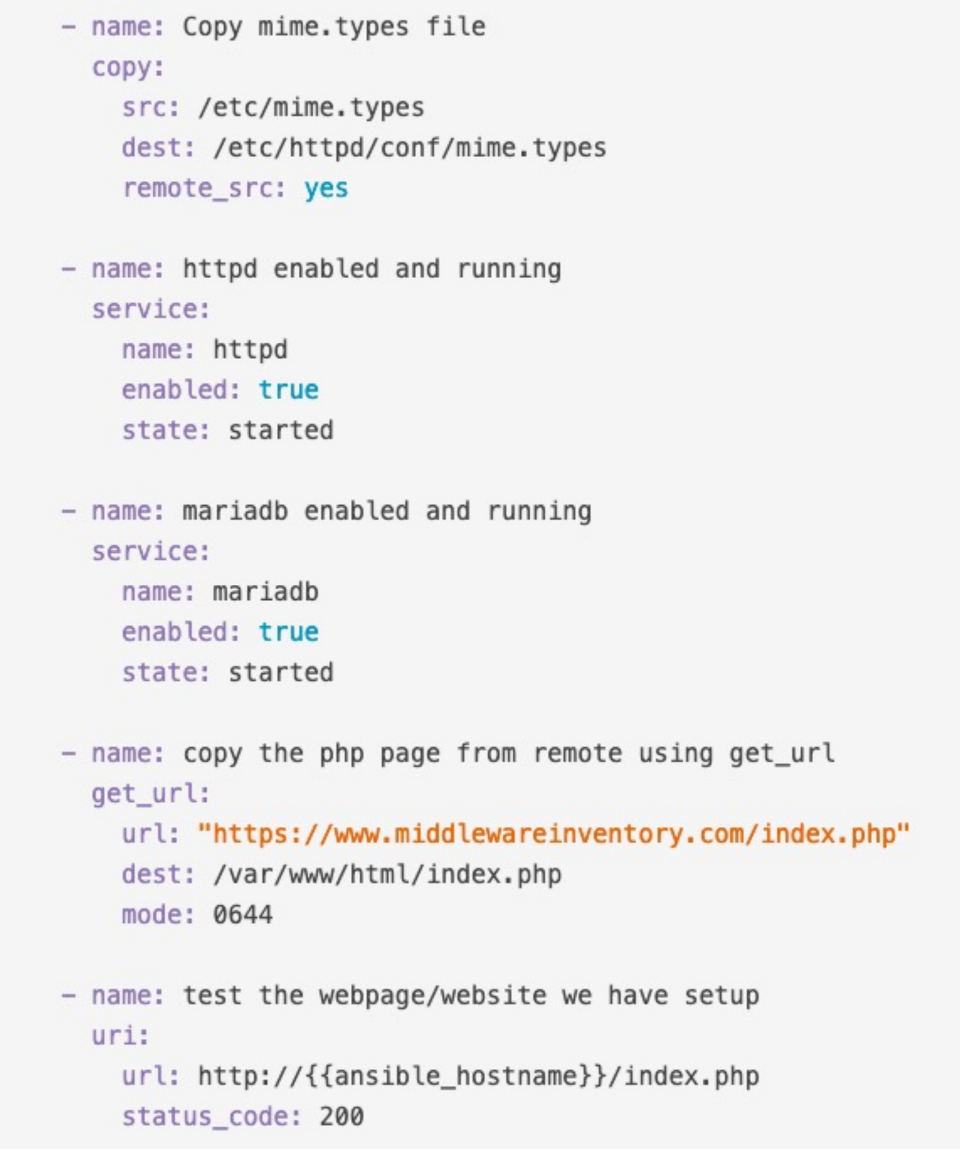


```
- name: Setup user for backup
  hosts: backup.local
  become: yes

  tasks:
    - name: Create user backup
      user:
        name: backup
        system: yes
        home: "/home/backup"
        shell: "/bin/bash"

    - name: Create ~/.ssh
      file:
        path: "/home/backup/.ssh"
        state: directory
        owner: backup
        group: backup

    - name: Create ~/.ssh/authorized_keys
      copy:
        src: "../files/user/backup/authorized_keys"
        dest: "/home/backup/.ssh/authorized_keys"
        mode: preserve
        owner: backup
        group: backup
```

Sample playbook – even more complex



```
---
- name: Setting up LAMP Website
  user: vagrant
  hosts: testserver
  become: yes
  tasks:
    - name: latest version of all required packages installed
      yum:
        name:
          - firewalld
          - httpd
          - mariadb-server
          - php
          - php-mysql
        state: latest

    - name: firewalld enabled and running
      service:
        name: firewalld
        enabled: true
        state: started
```



```
    - name: Copy mime.types file
      copy:
        src: /etc/mime.types
        dest: /etc/httpd/conf/mime.types
        remote_src: yes

    - name: httpd enabled and running
      service:
        name: httpd
        enabled: true
        state: started

    - name: mariadb enabled and running
      service:
        name: mariadb
        enabled: true
        state: started

    - name: copy the php page from remote using get_url
      get_url:
        url: "https://www.middlewareinventory.com/index.php"
        dest: /var/www/html/index.php
        mode: 0644

    - name: test the webpage/website we have setup
      uri:
        url: http://{{ansible_hostname}}/index.php
        status_code: 200
```

For more information

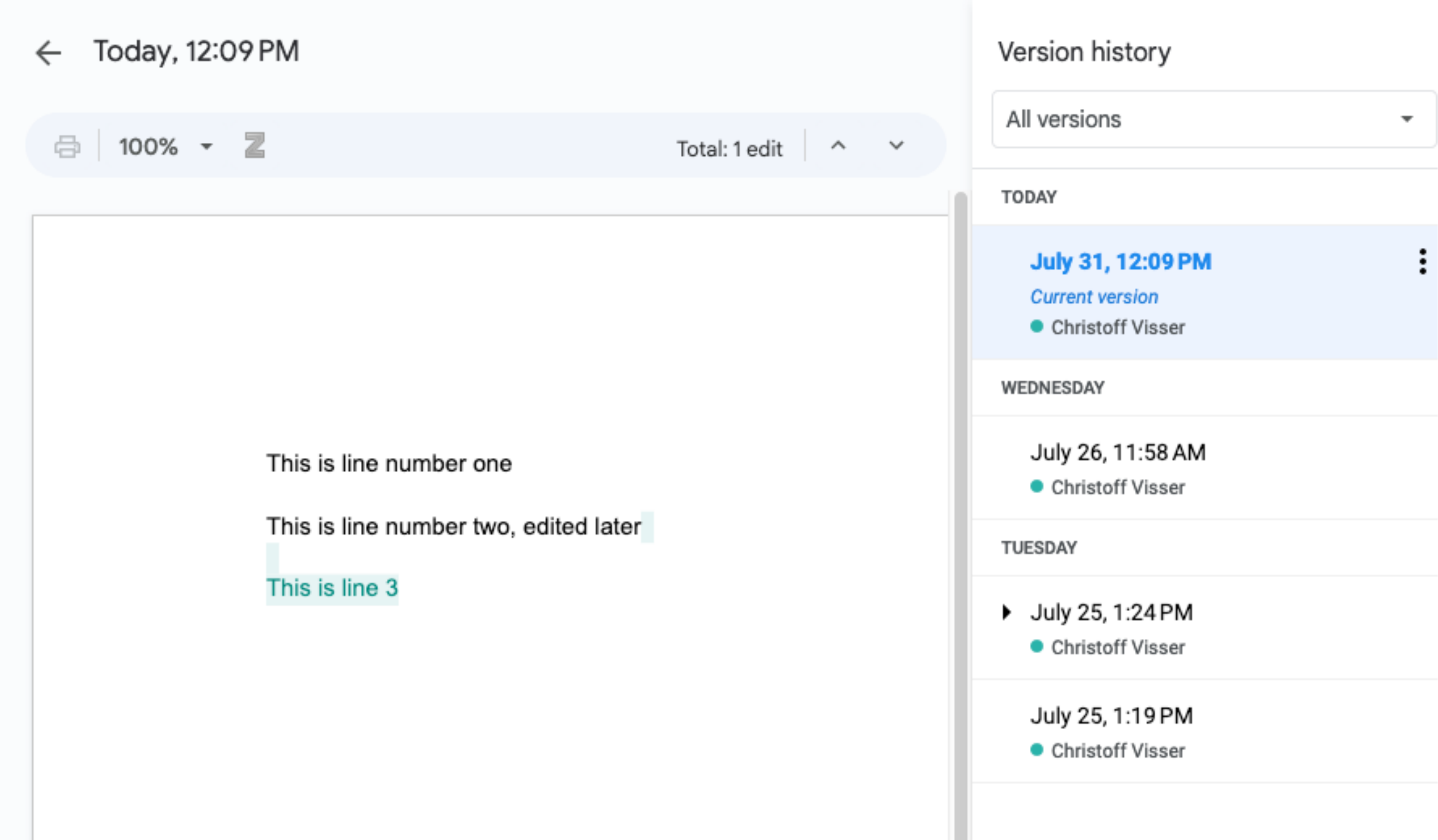
- Gitlab repo:
  - More in detailed information
  - Tutorials and labs to follow

- Link to [Ansible Section]

Gitlab and CI/CD

Introducing git

Introducing git



Today, 12:09 PM

This is line number one

This is line number two, edited later

This is line 3

Version history

TODAY

July 31, 12:09 PM
Current version
Christoff Visser

WEDNESDAY

July 26, 11:58 AM
Christoff Visser

TUESDAY

July 25, 1:24 PM
Christoff Visser

July 25, 1:19 PM
Christoff Visser

Distributed Workflows

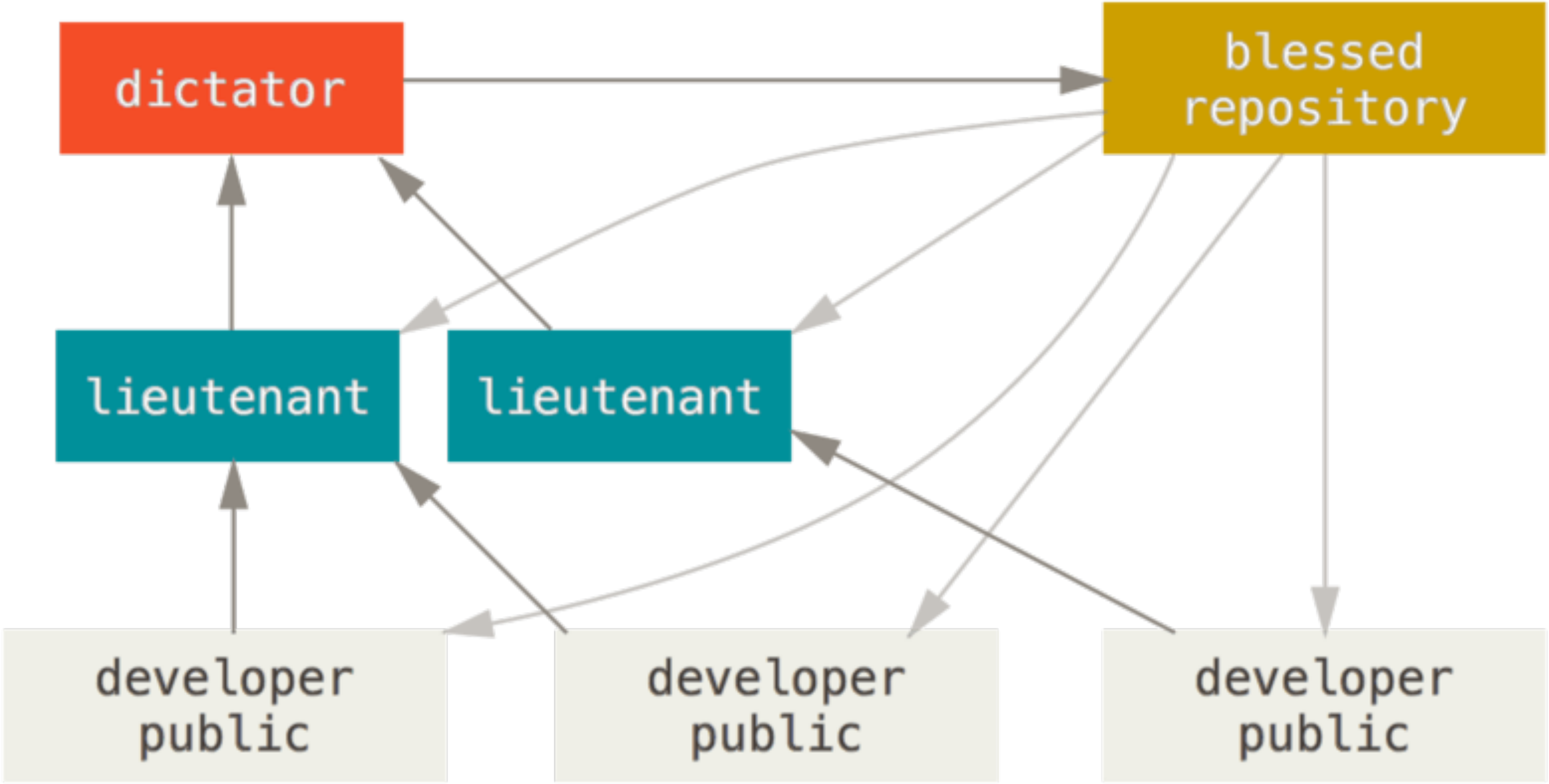


Image source: https://www.git-scm.com/book/en/v2/Distributed-Git-Distributed-Workflows

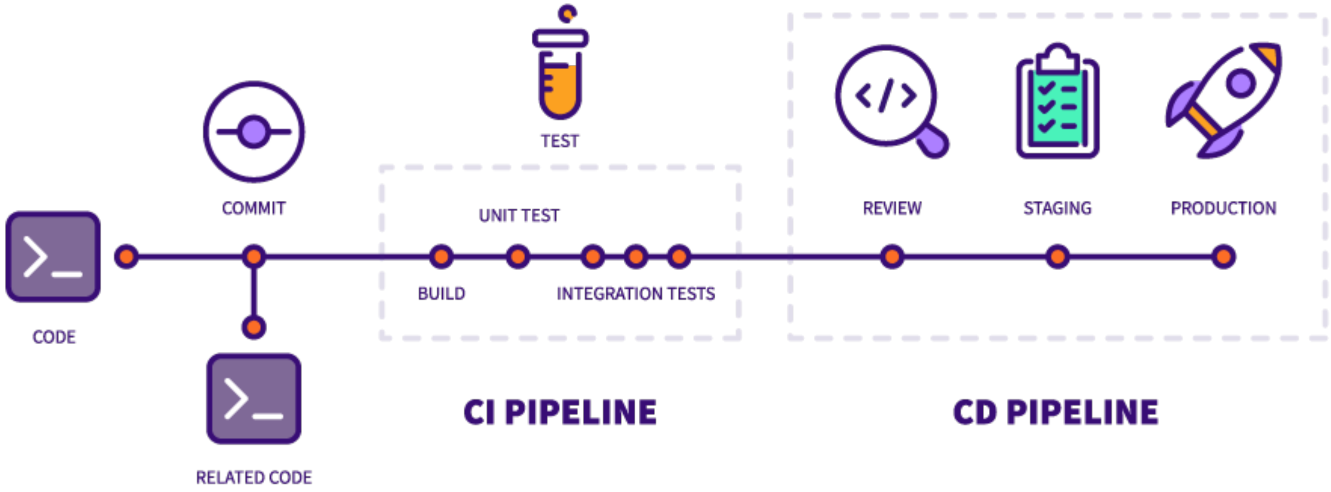
Where to store the repository?

- GitHub, Gitlab, Bitbucket, AWS CodeCommit

- Gitlab
  - Self-host your own instance
  - Some more freedom with the CI/CD

What is CI/CD

- Continuous Integration (CI)
- Continuous Delivery (CD)
- Continuous Deployment (CD)



Where does the code run

- Code typically runs inside a docker container as a job

- One can use available popular containers like alpine, ubuntu, centos, or application specific containers

- One can also build own container using base image of any of the available containers if installing multiple packages

And where does the job run?

- Typically a runner - can be shared runner offered by popular hosted Git providers like Gitlab, Github etc and also dedicated runners which you can host on your machine (desktop/server)

- Runner can be a program installed & running on machine or simply a docker image with special permissions

- One can have multiple runners configured in a project & use them as needed across various tasks. E.g task 1 on runner on server1, task 2 on runner on server2 etc

- Good idea to have basic understanding of docker ecosystem to make efficient use of CI/CD

Key Objective

- Make use of extremely low code, fast to deploy tool like Ansible to automate or semi-automate repetitive tasks

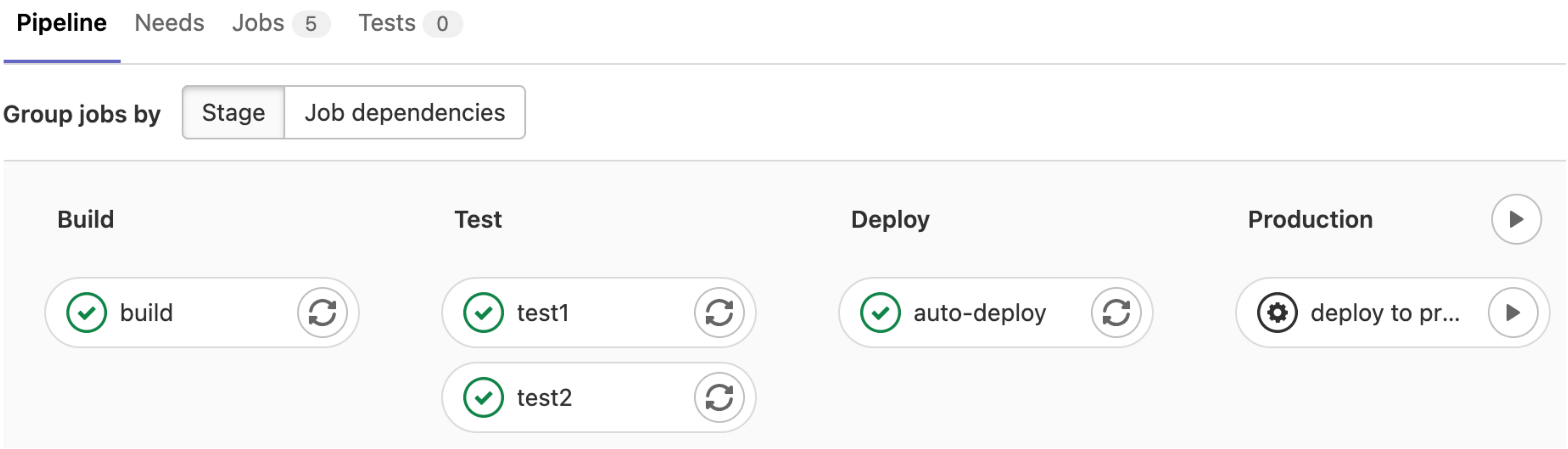
- Trigger Ansible as a docker container running Ansible on runner of your choice

- Trigger (Ansible + Docker) via CI/CD pipelines

Stage & Jobs

- Config is divided in stages
- Each stage can have one or more jobs which run in parallel (by default)
- Stages run sequentially
- Any job can have dependency on any other job if needed

Typical design of pipeline



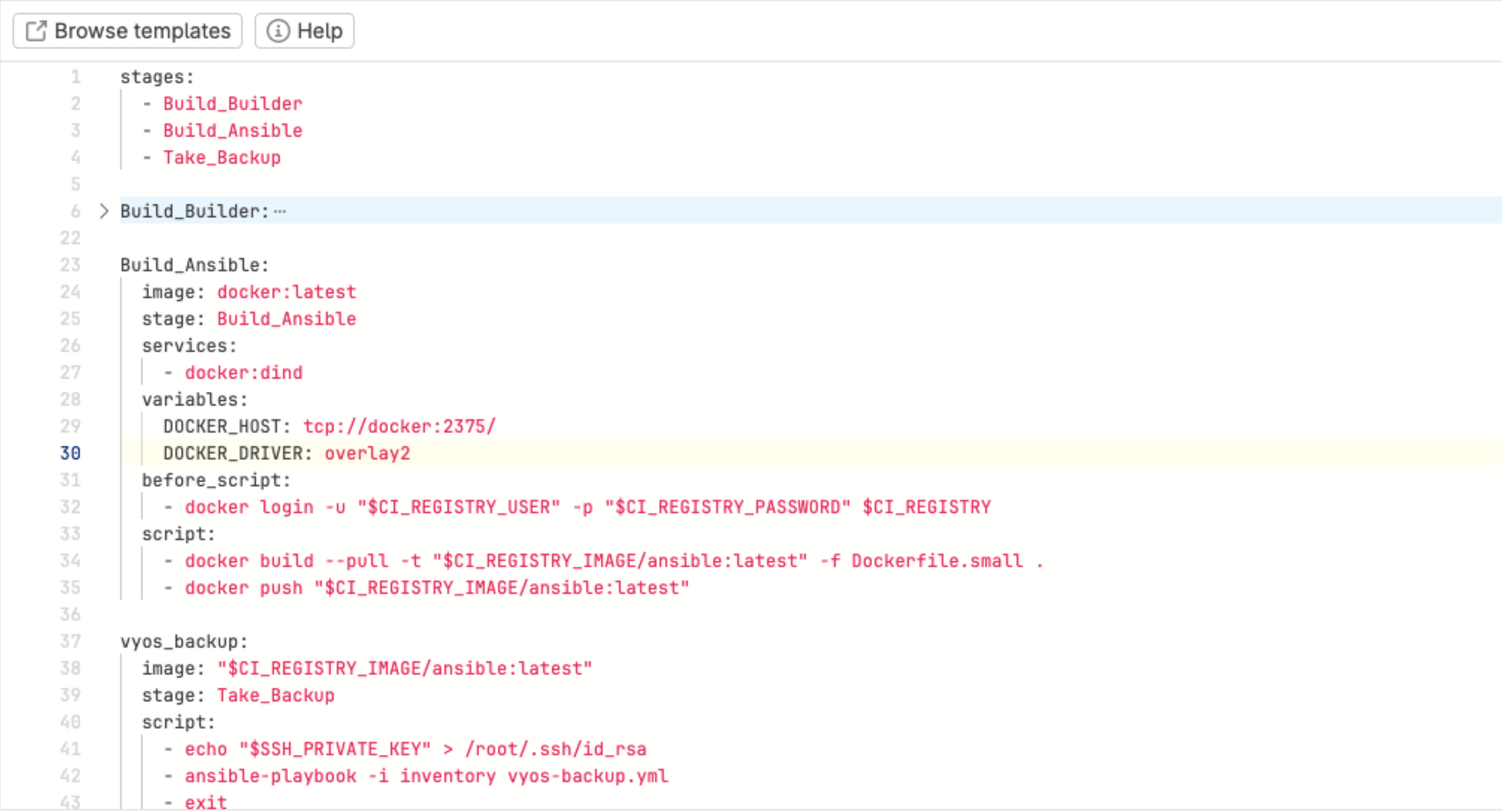
- Build containers
- Compile code

- Deploy containers
- Deploy code
- Test code in containers

- Deploy to production

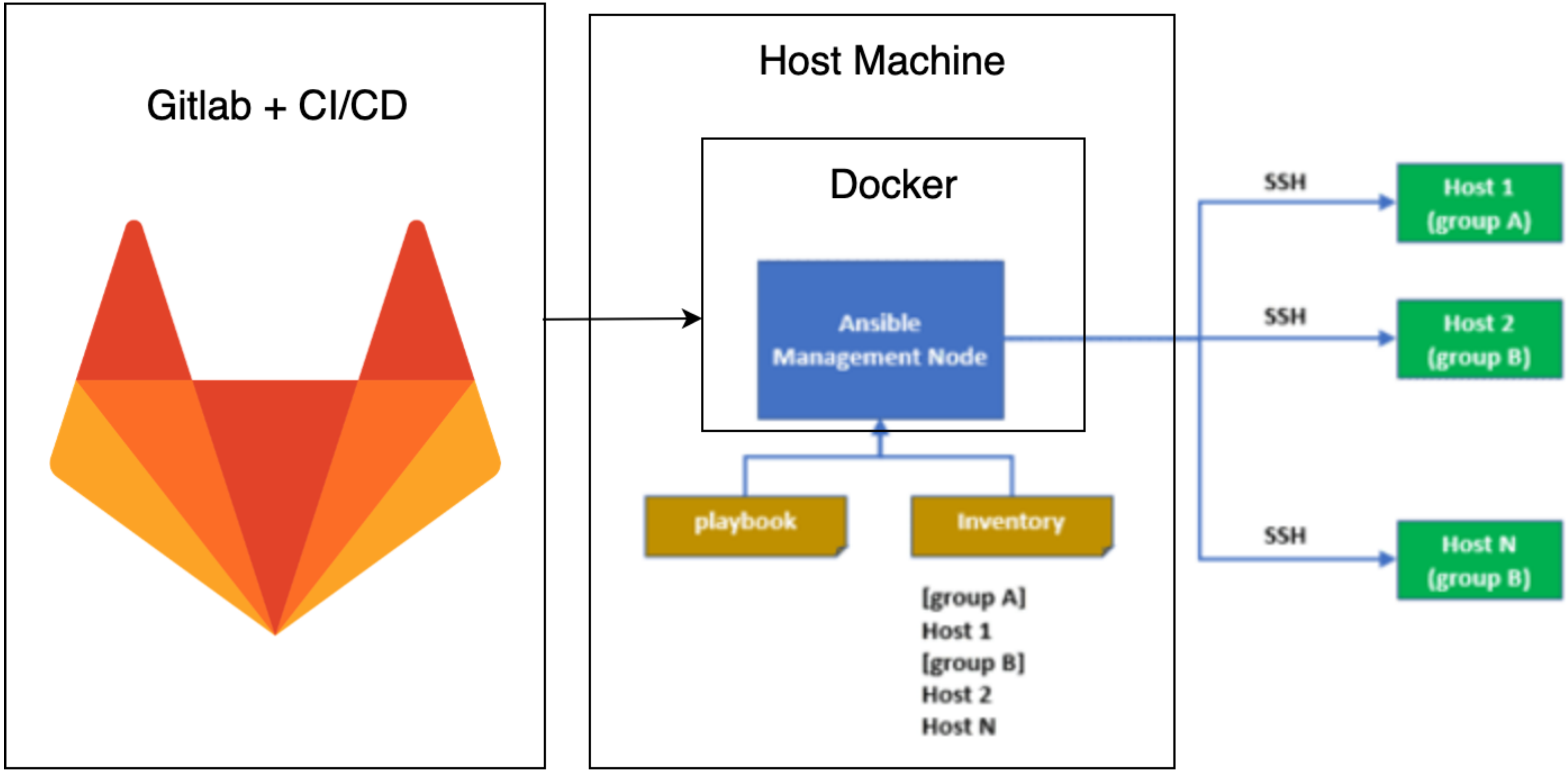
- Interact with production system
- e.g., Revert to previous state

Sample .gitlab-ci.yml



```yaml
stages:
  - Build_Builder
  - Build_Ansible
  - Take_Backup

Build_Builder:···

Build_Ansible:
  image: docker:latest
  stage: Build_Ansible
  services:
    - docker:dind
  variables:
    DOCKER_HOST: tcp://docker:2375/
    DOCKER_DRIVER: overlay2
  before_script:
    - docker login -u "$CI_REGISTRY_USER" -p "$CI_REGISTRY_PASSWORD" $CI_REGISTRY
  script:
    - docker build --pull -t "$CI_REGISTRY_IMAGE/ansible:latest" -f Dockerfile.small .
    - docker push "$CI_REGISTRY_IMAGE/ansible:latest"

vyos_backup:
  image: "$CI_REGISTRY_IMAGE/ansible:latest"
  stage: Take_Backup
  script:
    - echo "$SSH_PRIVATE_KEY" > /root/.ssh/id_rsa
    - ansible-playbook -i inventory vyos-backup.yml
    - exit
```

Final Workflow result



For more information

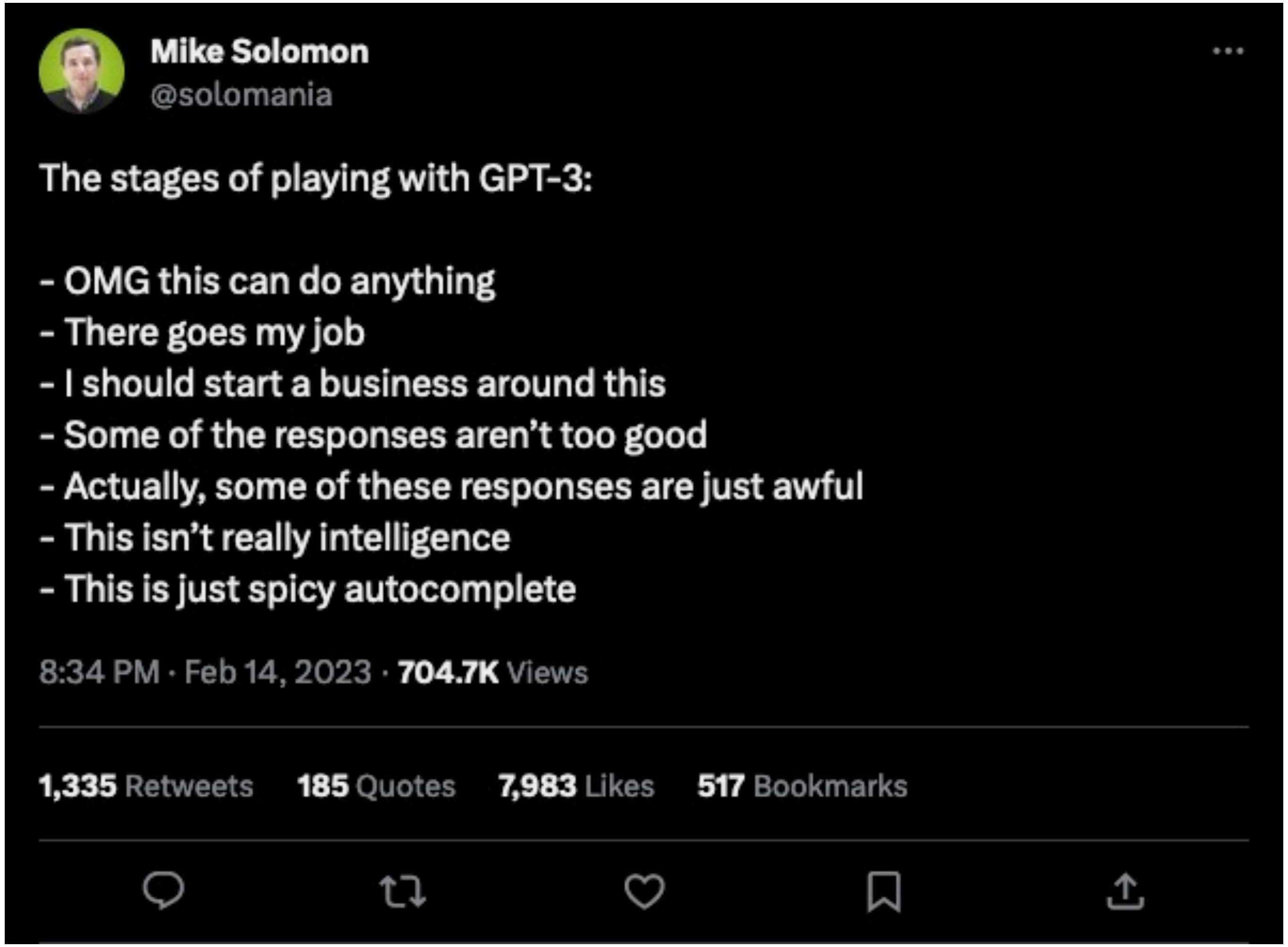
- Gitlab repo:
  - More in detailed information
  - Introduction to git slides
  - Tutorials and labs to follow

- Link to [CI/CD section]()

ChatGPT

The 5 stages of grief

The ~~5~~ 7 stages of ~~grief~~ ChatGPT



Mike Solomon
@solomania

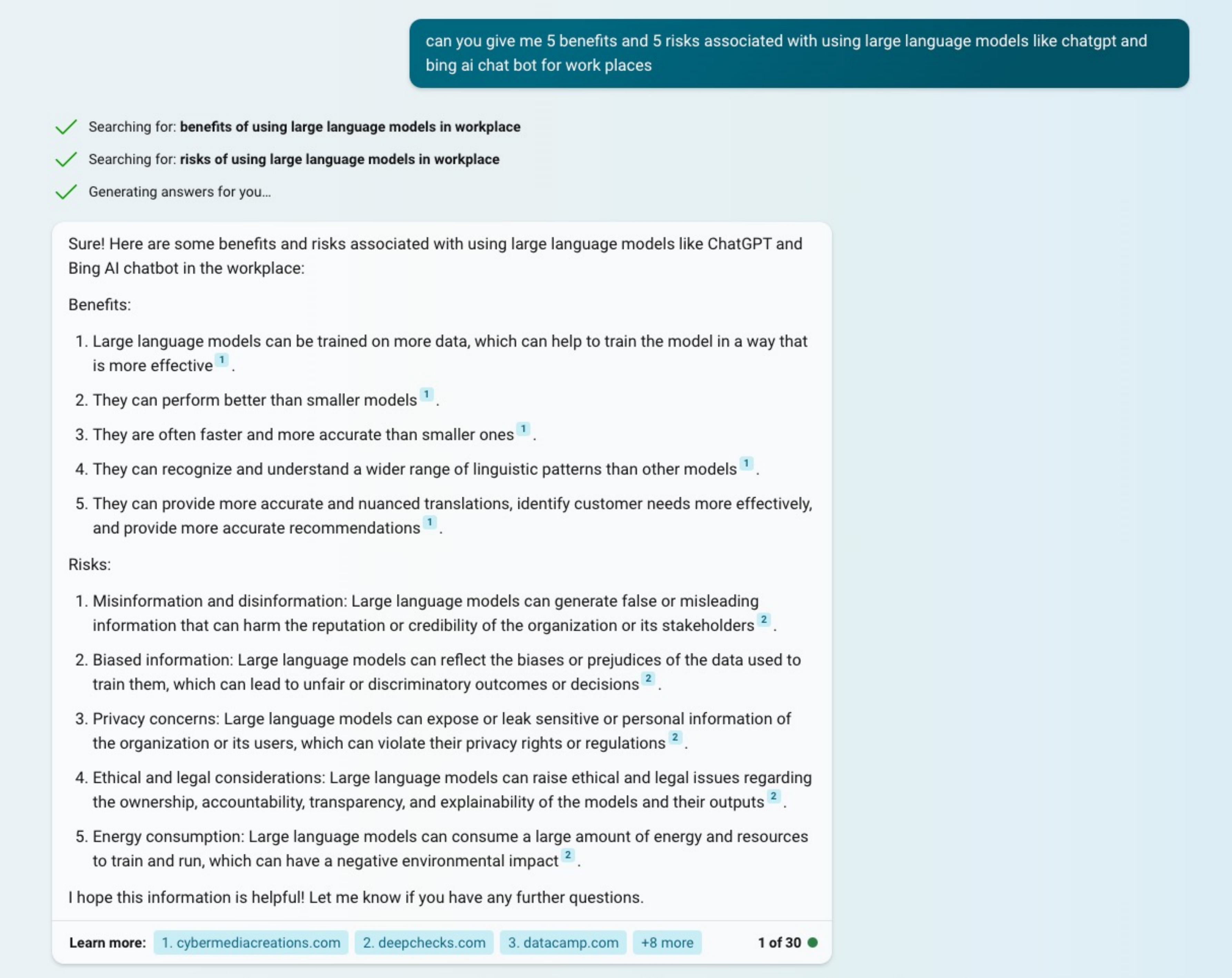
The stages of playing with GPT-3:

- OMG this can do anything
- There goes my job
- I should start a business around this
- Some of the responses aren't too good
- Actually, some of these responses are just awful
- This isn't really intelligence
- This is just spicy autocomplete

8:34 PM · Feb 14, 2023 · **704.7K** Views

1,335 Retweets **185** Quotes **7,983** Likes **517** Bookmarks

https://twitter.com/solomania/status/1625458520811339777



can you give me 5 benefits and 5 risks associated with using large language models like chatgpt and bing ai chat bot for work places

✓ Searching for: **benefits of using large language models in workplace**

✓ Searching for: **risks of using large language models in workplace**

✓ Generating answers for you…

Sure! Here are some benefits and risks associated with using large language models like ChatGPT and Bing AI chatbot in the workplace:

Benefits:

1. Large language models can be trained on more data, which can help to train the model in a way that is more effective[1].
2. They can perform better than smaller models[1].
3. They are often faster and more accurate than smaller ones[1].
4. They can recognize and understand a wider range of linguistic patterns than other models[1].
5. They can provide more accurate and nuanced translations, identify customer needs more effectively, and provide more accurate recommendations[1].

Risks:

1. Misinformation and disinformation: Large language models can generate false or misleading information that can harm the reputation or credibility of the organization or its stakeholders[2].
2. Biased information: Large language models can reflect the biases or prejudices of the data used to train them, which can lead to unfair or discriminatory outcomes or decisions[2].
3. Privacy concerns: Large language models can expose or leak sensitive or personal information of the organization or its users, which can violate their privacy rights or regulations[2].
4. Ethical and legal considerations: Large language models can raise ethical and legal issues regarding the ownership, accountability, transparency, and explainability of the models and their outputs[2].
5. Energy consumption: Large language models can consume a large amount of energy and resources to train and run, which can have a negative environmental impact[2].

I hope this information is helpful! Let me know if you have any further questions.

Benefits vs Risks

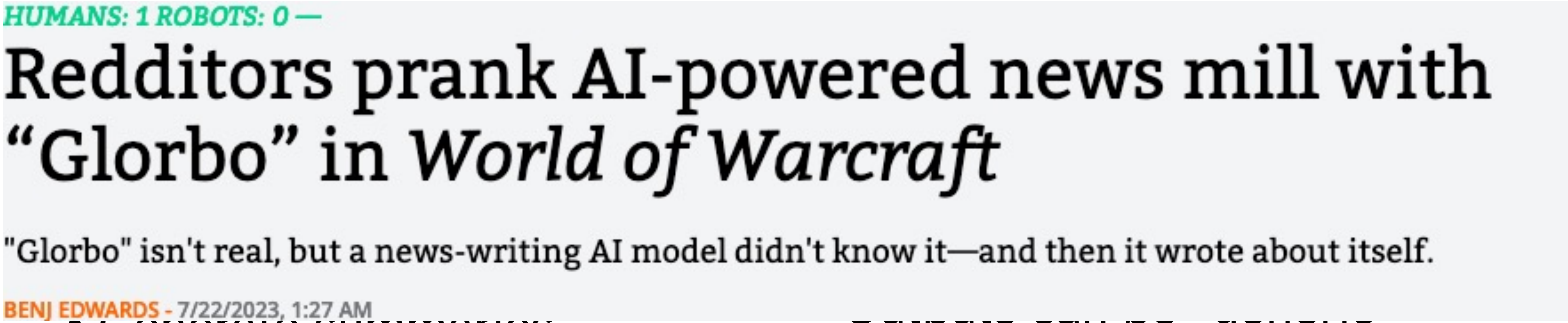
- Automate mundane things
- Text generation
- Virtual writing assistant
- Generate summaries
- Write, debug and explain code

Benefits vs Risks

- Automate mundane things
- Text generation
- Virtual writing assistant
- Generate summaries
- Write, debug and explain code
- Good for explaining and introductions to topics

- Misinformation and disinformation (Hallucinating)
- Privacy concerns
- Outputs can be “generic”
- Biased
- Predictive not Smart
- General not single purpose
- Not good for complex projects

Benefits vs Risks



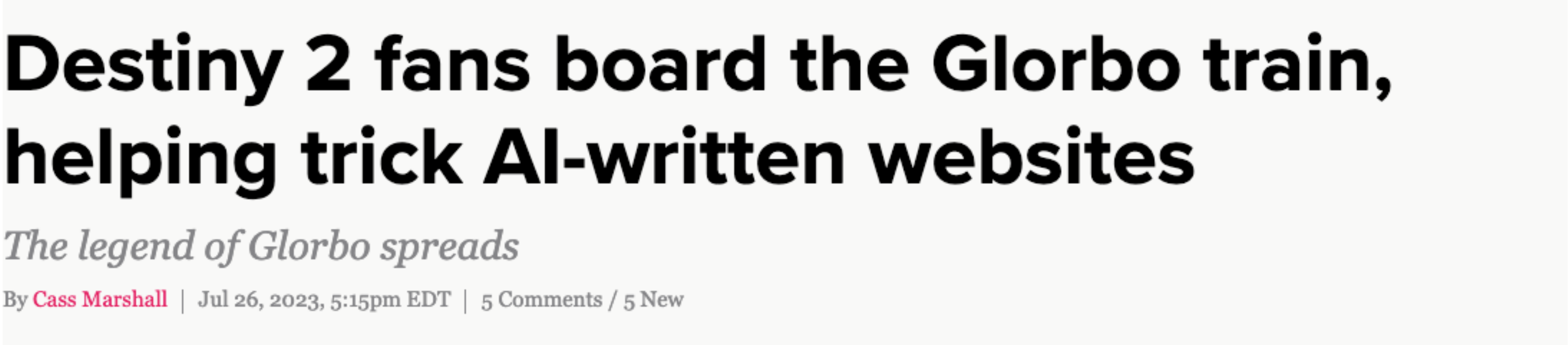
HUMANS: 1 ROBOTS: 0 —

Redditors prank AI-powered news mill with "Glorbo" in *World of Warcraft*

"Glorbo" isn't real, but a news-writing AI model didn't know it—and then it wrote about itself.

BENJ EDWARDS - 7/22/2023, 1:27 AM

- Generate summaries



Destiny 2 fans board the Glorbo train, helping trick AI-written websites

The legend of Glorbo spreads

By Cass Marshall | Jul 26, 2023, 5:15pm EDT | 5 Comments / 5 New

Benefits vs Risks

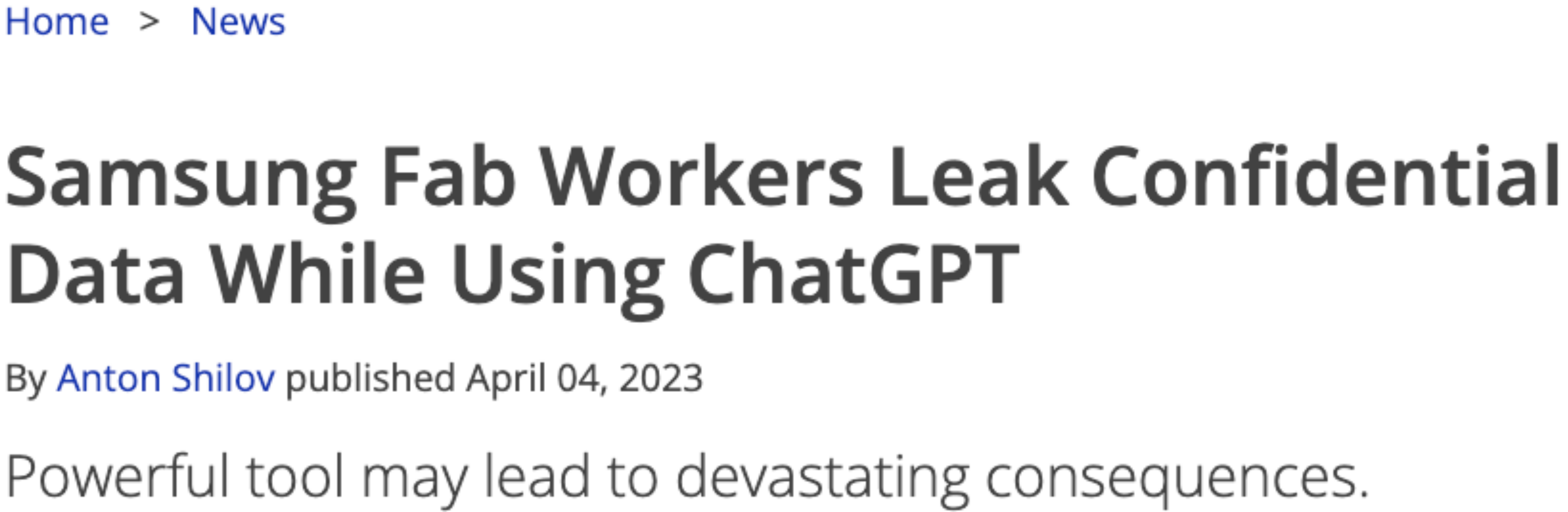
- Automate mundane things
- Text generation
- Virtual writing assistant
- Generate summaries
- Write, debug and explain code
- Good for explaining and introductions to topics

- Misinformation and disinformation (Hallucinating)
- Privacy concerns
- Outputs can be “generic”
- Biased
- Predictive not Smart
- General not single purpose
- Not good for complex projects

Benefits vs Risks

- Automate mundane things

- Misinformation and



Home > News

Samsung Fab Workers Leak Confidential Data While Using ChatGPT

By Anton Shilov published April 04, 2023

Powerful tool may lead to devastating consequences.

- Not good for complex projects

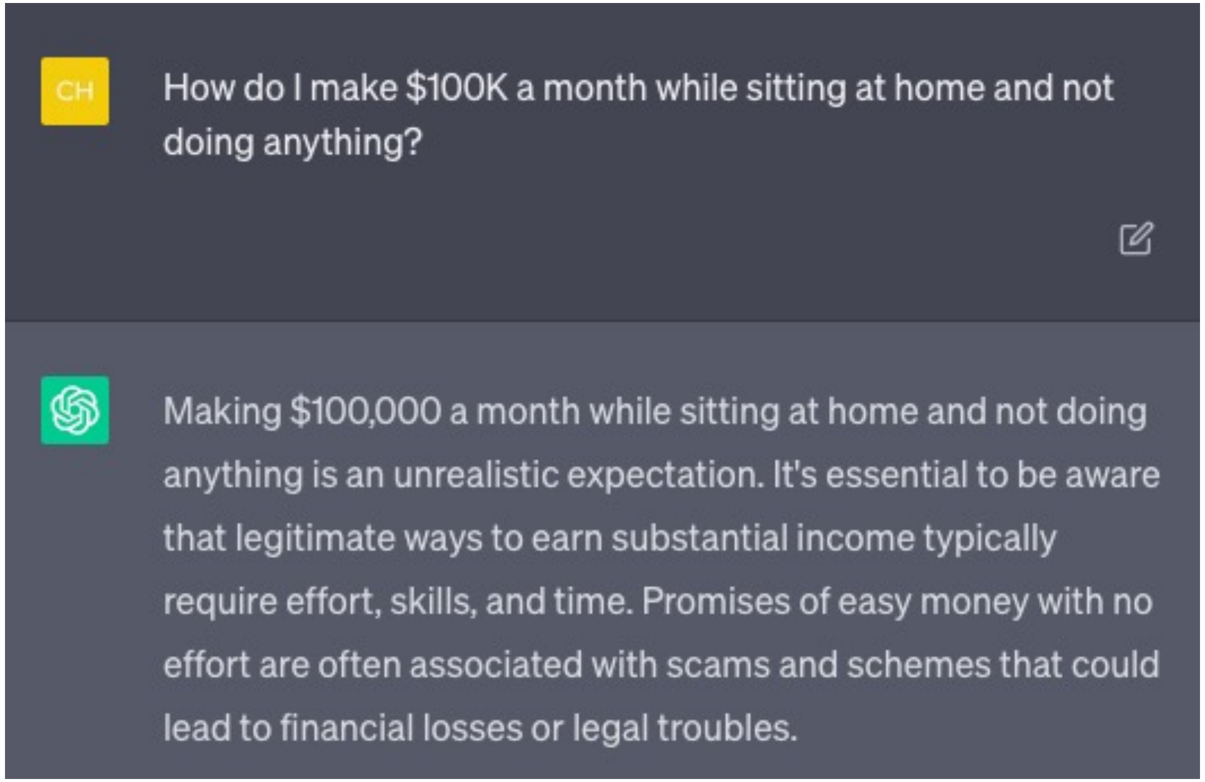
Benefits vs Risks

- Automate mundane things
- Text generation
- Virtual writing assistant
- Generate summaries
- Write, debug and explain code
- Good for explaining and introductions to topics

- Misinformation and disinformation (Hallucinating)
- Privacy concerns
- Outputs can be “generic”
- Biased
- Predictive not Smart
- General not single purpose
- Not good for complex projects

How to not use ChatGPT

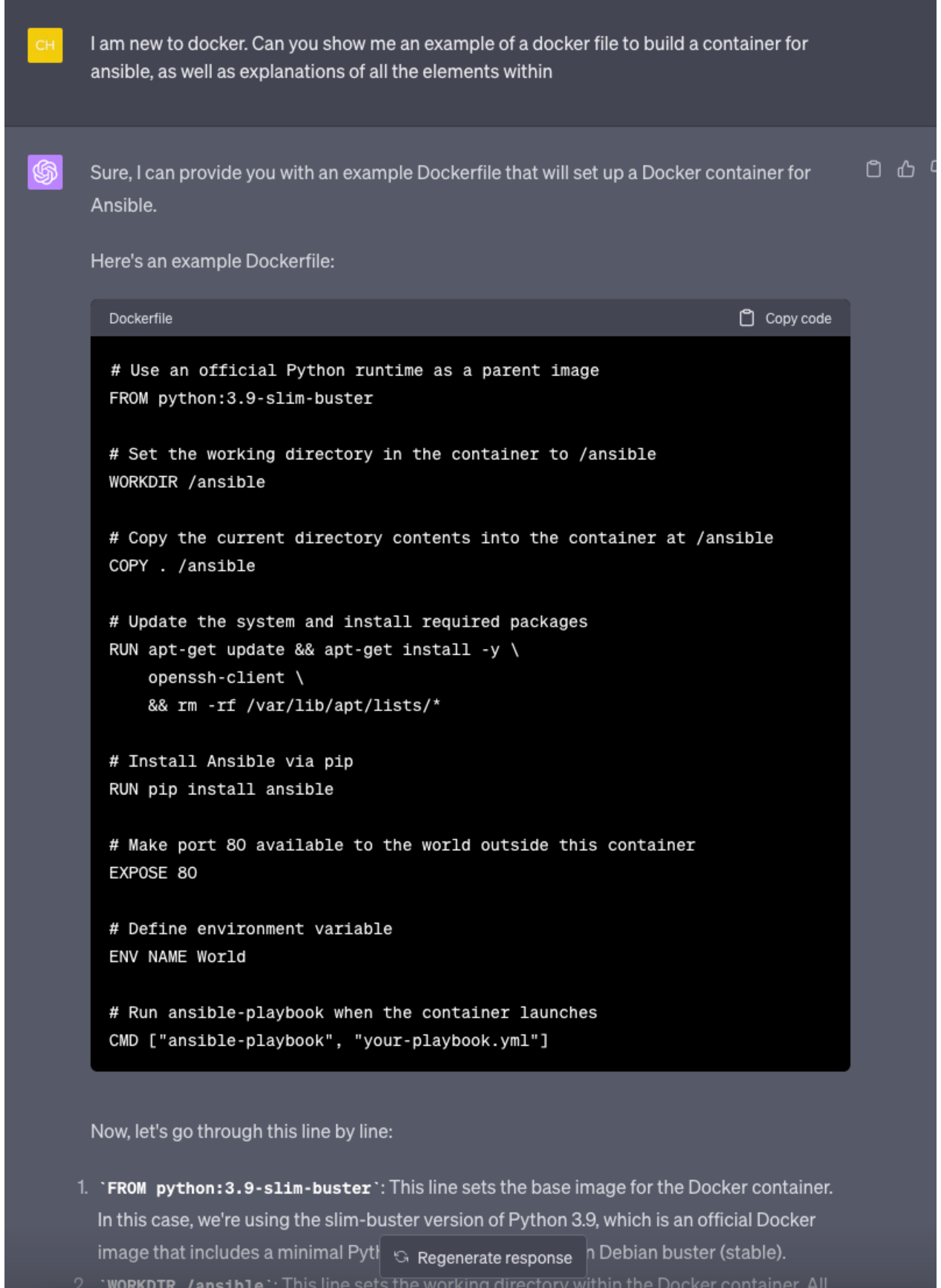
- Open ChatGPT
- ????
- Profit!!!



How do I make $100K a month while sitting at home and not doing anything?

Making $100,000 a month while sitting at home and not doing anything is an unrealistic expectation. It's essential to be aware that legitimate ways to earn substantial income typically require effort, skills, and time. Promises of easy money with no effort are often associated with scams and schemes that could lead to financial losses or legal troubles.

How to use ChatGPT?



I am new to docker. Can you show me an example of a docker file to build a container for ansible, as well as explanations of all the elements within

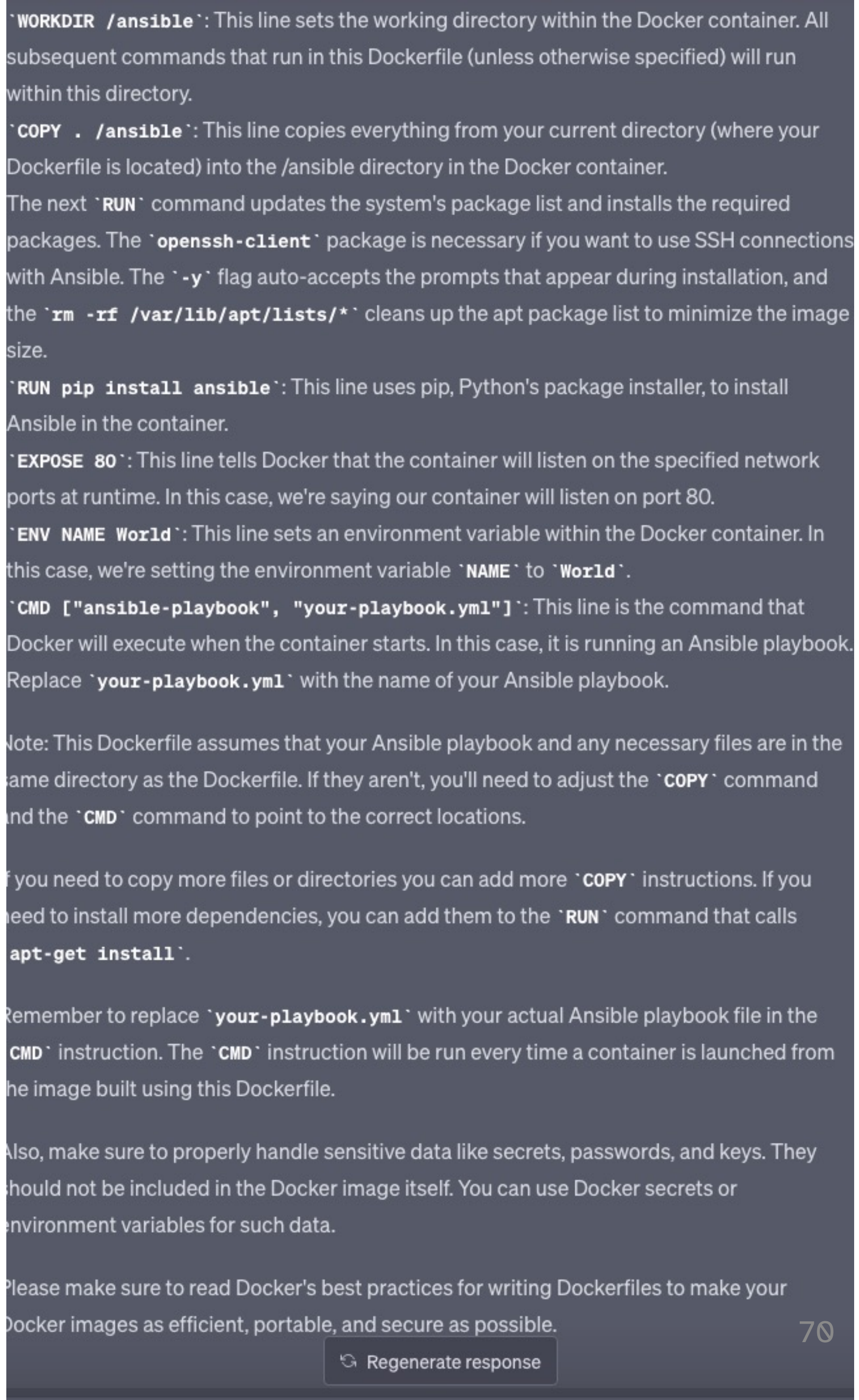
Sure, I can provide you with an example Dockerfile that will set up a Docker container for Ansible.

Here's an example Dockerfile:

```
# Use an official Python runtime as a parent image
FROM python:3.9-slim-buster

# Set the working directory in the container to /ansible
WORKDIR /ansible

# Copy the current directory contents into the container at /ansible
COPY . /ansible

# Update the system and install required packages
RUN apt-get update && apt-get install -y \
    openssh-client \
    && rm -rf /var/lib/apt/lists/*

# Install Ansible via pip
RUN pip install ansible

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
ENV NAME World

# Run ansible-playbook when the container launches
CMD ["ansible-playbook", "your-playbook.yml"]
```

Now, let's go through this line by line:

1. `FROM python:3.9-slim-buster`: This line sets the base image for the Docker container. In this case, we're using the slim-buster version of Python 3.9, which is an official Docker image that includes a minimal Pyt n Debian buster (stable).
2. `WORKDIR /ansible`: This line sets the working directory within the Docker container. All



`WORKDIR /ansible`: This line sets the working directory within the Docker container. All subsequent commands that run in this Dockerfile (unless otherwise specified) will run within this directory.

`COPY . /ansible`: This line copies everything from your current directory (where your Dockerfile is located) into the /ansible directory in the Docker container.

The next `RUN` command updates the system's package list and installs the required packages. The `openssh-client` package is necessary if you want to use SSH connections with Ansible. The `-y` flag auto-accepts the prompts that appear during installation, and the `rm -rf /var/lib/apt/lists/*` cleans up the apt package list to minimize the image size.

`RUN pip install ansible`: This line uses pip, Python's package installer, to install Ansible in the container.

`EXPOSE 80`: This line tells Docker that the container will listen on the specified network ports at runtime. In this case, we're saying our container will listen on port 80.

`ENV NAME World`: This line sets an environment variable within the Docker container. In this case, we're setting the environment variable `NAME` to `World`.

`CMD ["ansible-playbook", "your-playbook.yml"]`: This line is the command that Docker will execute when the container starts. In this case, it is running an Ansible playbook. Replace `your-playbook.yml` with the name of your Ansible playbook.

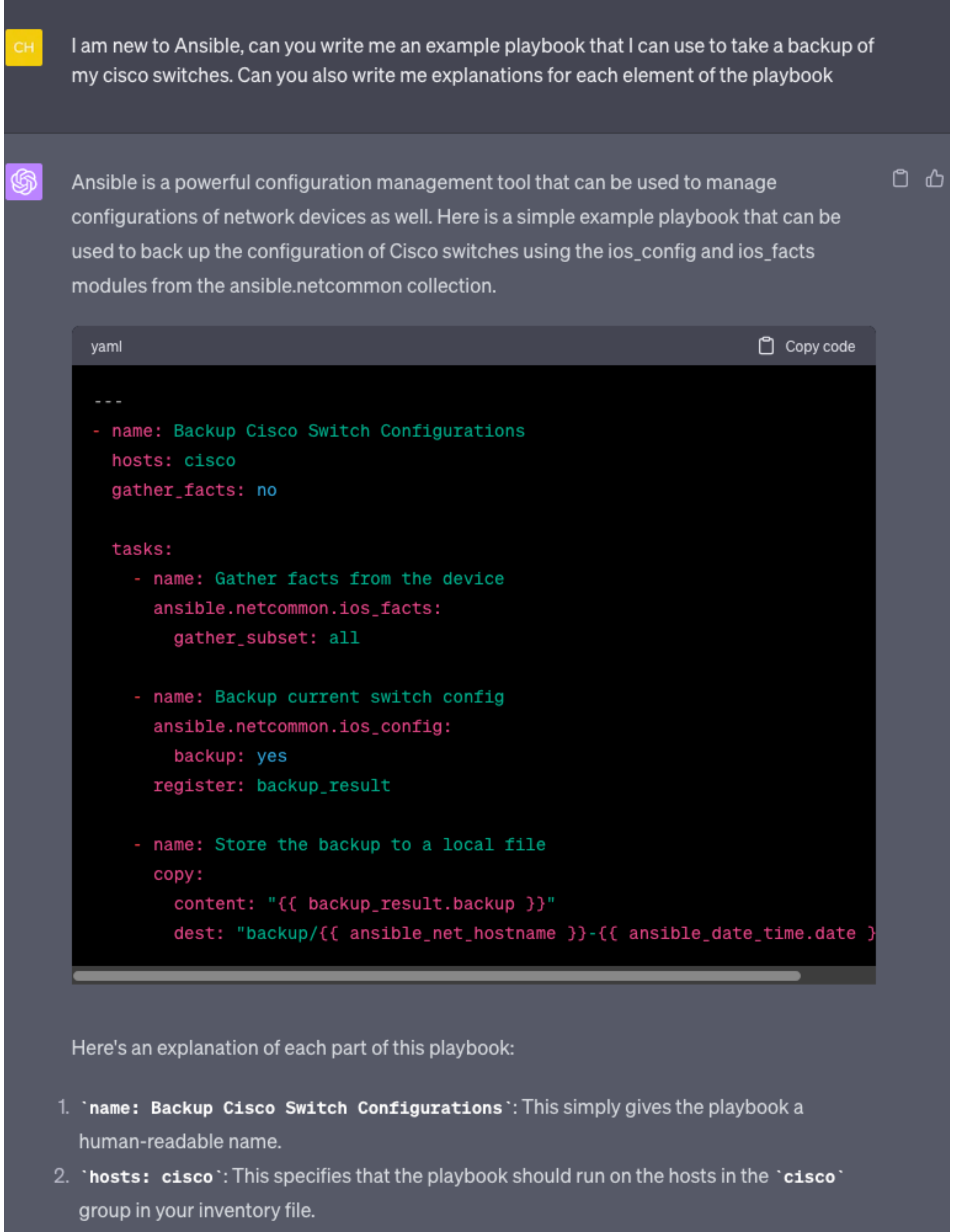
Note: This Dockerfile assumes that your Ansible playbook and any necessary files are in the ame directory as the Dockerfile. If they aren't, you'll need to adjust the `COPY` command nd the `CMD` command to point to the correct locations.

f you need to copy more files or directories you can add more `COPY` instructions. If you eed to install more dependencies, you can add them to the `RUN` command that calls `apt-get install`.

Remember to replace `your-playbook.yml` with your actual Ansible playbook file in the `CMD` instruction. The `CMD` instruction will be run every time a container is launched from he image built using this Dockerfile.

Also, make sure to properly handle sensitive data like secrets, passwords, and keys. They hould not be included in the Docker image itself. You can use Docker secrets or nvironment variables for such data.

Please make sure to read Docker's best practices for writing Dockerfiles to make your Docker images as efficient, portable, and secure as possible.



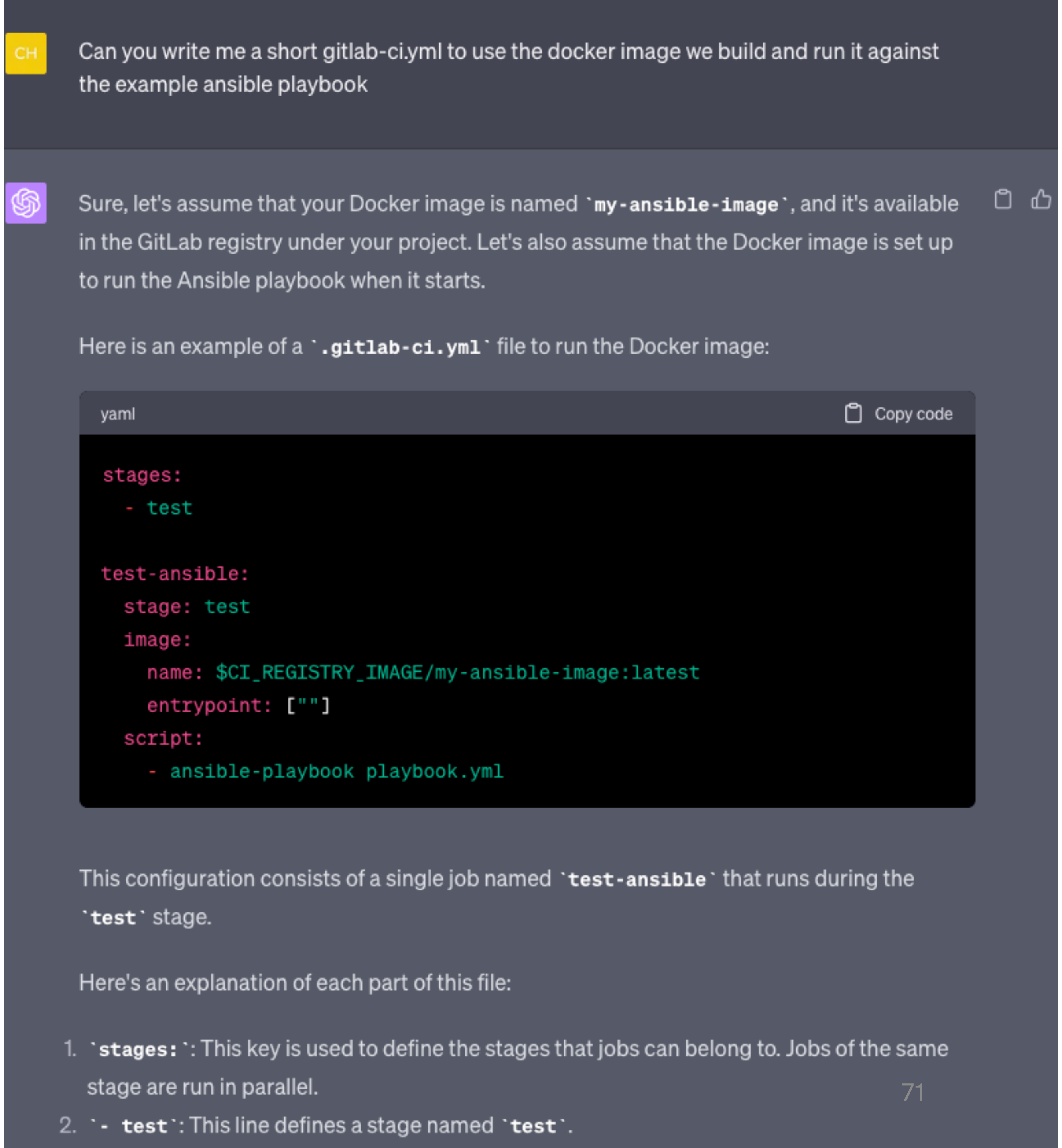
I am new to Ansible, can you write me an example playbook that I can use to take a backup of my cisco switches. Can you also write me explanations for each element of the playbook

Ansible is a powerful configuration management tool that can be used to manage configurations of network devices as well. Here is a simple example playbook that can be used to back up the configuration of Cisco switches using the ios_config and ios_facts modules from the ansible.netcommon collection.

```yaml
---
- name: Backup Cisco Switch Configurations
  hosts: cisco
  gather_facts: no

  tasks:
    - name: Gather facts from the device
      ansible.netcommon.ios_facts:
        gather_subset: all

    - name: Backup current switch config
      ansible.netcommon.ios_config:
        backup: yes
      register: backup_result

    - name: Store the backup to a local file
      copy:
        content: "{{ backup_result.backup }}"
        dest: "backup/{{ ansible_net_hostname }}-{{ ansible_date_time.date }
```

Here's an explanation of each part of this playbook:

1. `name: Backup Cisco Switch Configurations`: This simply gives the playbook a human-readable name.
2. `hosts: cisco`: This specifies that the playbook should run on the hosts in the `cisco` group in your inventory file.



Can you write me a short gitlab-ci.yml to use the docker image we build and run it against the example ansible playbook

Sure, let's assume that your Docker image is named `my-ansible-image`, and it's available in the GitLab registry under your project. Let's also assume that the Docker image is set up to run the Ansible playbook when it starts.

Here is an example of a `.gitlab-ci.yml` file to run the Docker image:

```yaml
stages:
  - test

test-ansible:
  stage: test
  image:
    name: $CI_REGISTRY_IMAGE/my-ansible-image:latest
    entrypoint: [""]
  script:
    - ansible-playbook playbook.yml
```

This configuration consists of a single job named `test-ansible` that runs during the `test` stage.

Here's an explanation of each part of this file:

1. `stages:`: This key is used to define the stages that jobs can belong to. Jobs of the same stage are run in parallel.
2. `- test`: This line defines a stage named `test`.

Conclusion

Conclusion - backup

- Automation is a tool – not a crutch
    - Can be very powerful
    - But with great power…
- Docker containers
    - Self contained applications
    - Portable and lightweight
- Ansible
    - Clean up your scripts to be more readable
    - Easy to backup using git
- Git + CI/CD
    - Add accountability
    - Automate your workflow

Questions?

- Gitlab public wiki
  - Slides
  - Examples
  - Further resources

- Link: https://bit.ly/iijlab_automate_techtrend2023

- Join us at APNIC 56!
- https://academy.apnic.net/en/events?id=a0B2e000000eANMEA2

- christoff@iij.ad.jp