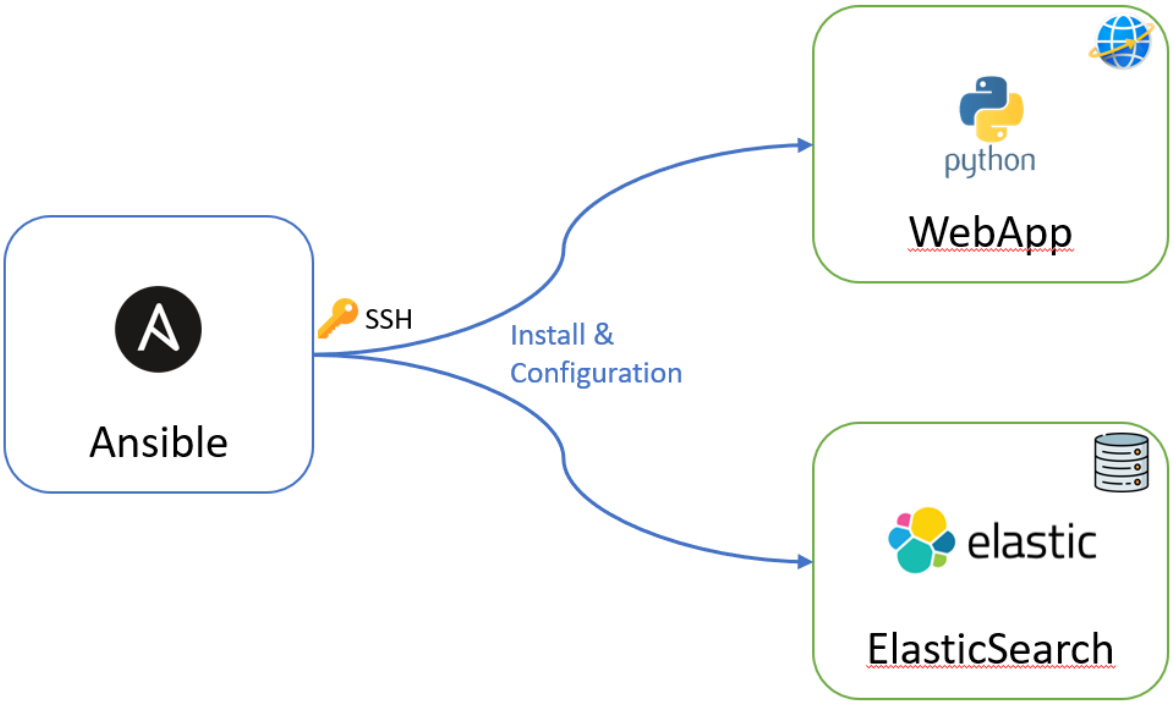
Ansible - IaC & Continuous Deployment

In this exercise, we will simulate the creation and configuration of a cluster of machines on which we will deploy:

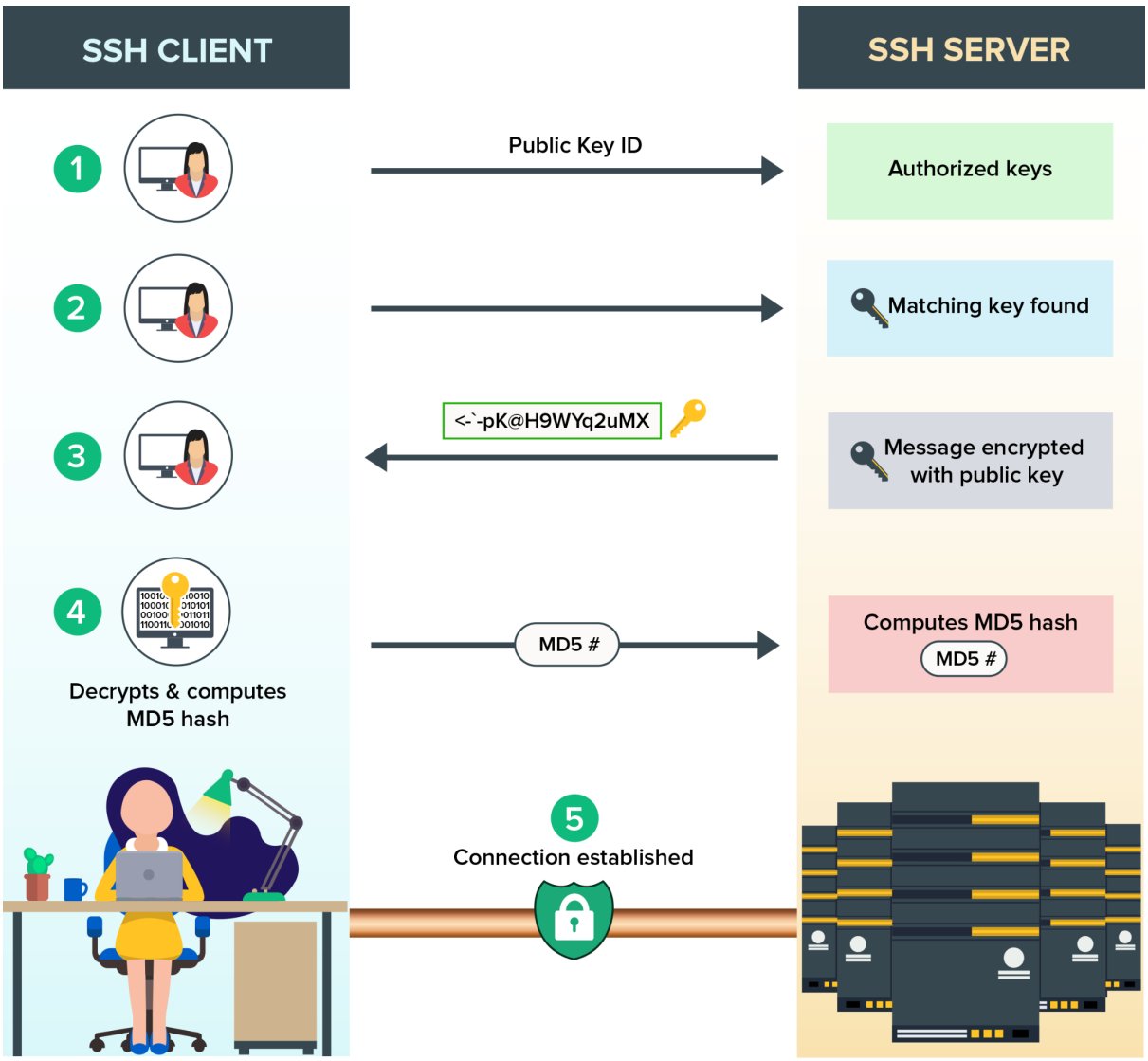
- Ansible, for deployments.
- An Elasticsearch Server, for search.
- A Python web application.

We'll use Docker to create three distinct containers:



Important notes :

- You'll need to generate or use a SSH RSA key. Remember you've already created one for Github.
- Docker is helpful for the current simulation. In real environments, you'd need real infrastructure (physical or at least VMs) for both Ansible and your servers. Ansible would typically be installed on your Jenkins (CI/CD) server.
- We refer to ubuntu images using the latest tag. This is not a good practice if we want to make our work reproductible. Feel free to use specific versions.



I - Using Ansible for IaC

A - Preparing a custom image for Ansible

- Create a directory for your Ansible Project.
- Create a subdirectory named “docker” and open it.
- Copy your id_rsa and id_rsa.pub keys in the subdirectory.
- Create a Dockerfile (named, for instance, ansible-dockerfile) with the following content instructions:

```
FROM ubuntu:latest
# Avoid prompts from apt
ENV DEBIAN_FRONTEND=noninteractive
# Install Ansible
RUN apt-get update && \
  apt-get install -y software-properties-common && \
```

```
  add-apt-repository --yes --update ppa:ansible/ansible && \
  apt-get install -y ansible

# Copy your private SSH key so your Ansible host can connect to the other containers
COPY id_rsa /root/.ssh/id_rsa
RUN chmod 600 /root/.ssh/id_rsa

# Clean up to reduce image size
RUN apt-get clean && \
  rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*

CMD ["/bin/bash"]
```

- Build your image :

```
docker build -t ansible-host:1.0 -f ansible-dockerfile .
```

B - Preparing a custom image for the servers

Since Ansible requires SSH connections, we will create a custom Ubuntu image with the following requirements:

- OpenSSH server must be installed (openssh-server package)
- Your public key's content must be added to ~/.ssh/authorized_keys file in your image. Note: your public key MUST be copied in your build context. Otherwise it won't be found.

```
$ cat Dockerfile
FROM ubuntu:latest
RUN apt-get update && apt-get install -y openssh-server
RUN mkdir -p /run/sshd
ADD id_rsa.pub /root/.ssh/authorized_keys
EXPOSE 22
CMD ["/usr/sbin/sshd","-D"]
$ docker build -t ubuntu-ssh:0.1 .
Successfully tagged ubuntu-ssh:0.1
$ docker run -d --name=testing ubuntu-ssh:0.1
$ docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' testing
172.17.0.4
```

Create an Ansible container and check you can connect to a server container using a SSH connection.

```
$ ssh root@172.17.0.4
root@4beadb541361:~#
```

It works !!

- Clean your environment by removing both containers.

C - Creating the nodes

Now we've prepared the Docker images, we will create four containers in a same network:

- A first one called "ansible"
  - Use the first image
  - Must join your custom network
  - Should mount a volume so it can access Ansible Project files for upcoming deployments.
- A second one called "elastic"
  - Use the servers image
  - Must join your custom network
- Two others called "app-dev" and "app-prod"
  - Use the servers image
  - Must join your custom network

You can either create a docker-compose file for this or simply create everything from the CLI.

Start your containers and check your Ansible container can connect to all the other containers using ssh and the container names.

ssh root@elastic-server

D - Preparing your infrastructure

Create an ansible project to prepare your environment. You're free to ask ChatGPT's help for the syntax, but please make sure to understand each step:

- Create an inventory file with three groups (app / elastic)
- Create a jdk role which installs the openjdk-11-jdk package
- Create a python role which installs python3, pip and two python libraries : flask+elasticsearch
- Create an elasticsearch role which downloads, extracts and runs Elasticsearch. This role will also download the movies dataset (https://storage.cloud.google.com/baitmbarek-dataops/movies-db-bulk.zip) and create an index containing this dataset if it doesn't already exist.
- Create a playbook which:
  - runs the jdk and elasticsearch roles on elasticsearch group nodes
  - runs the python role on app group nodes

Once everything's defined, run your playbooks to prepare the containers by installing.
Example command:
$ ansible-playbook -i inventory.ini elastic-playbook.yml

II - Using Ansible for Continuous Delivery

Now you've set up the infrastructure, we will focus on delivering applicative components on "app" nodes from Jenkins pipelines.

- Create a Jenkins image with Ansible installed and run a new container.
- Fork the following repo (if not already done) : https://github.com/baitmbarek/dataops-pyhton-movie-search
- Create a Jenkins pipeline which pulls the repository and invokes a new playbook which deploys the code in the selected app node (dev or prod), depending on a Jenkins parameter (you can use Ansible environment variables for this).
- The playbook should restart the services after each deployment.
- Make changes on the source code, run the pipeline, make sure your modifications are effective.

III - Dealing with secrets

- Install ansible-vault
- Create a file (ansible-vault create)
- Edit it and create two keys : dev_password and prod_password. Choose a value for each one.
- When deploying the application using your pipeline and your playbook, add a file named elastic-secret with chmod 700 containing the targeted environment password