Tutorial Objectives

1. Revision - Docker
2. Revision - GitHub Actions
3. Add deploy-job to workflow
    a. Configure EC2 instance as a GitHub Runner
    b. Trigger the workflow

Docker

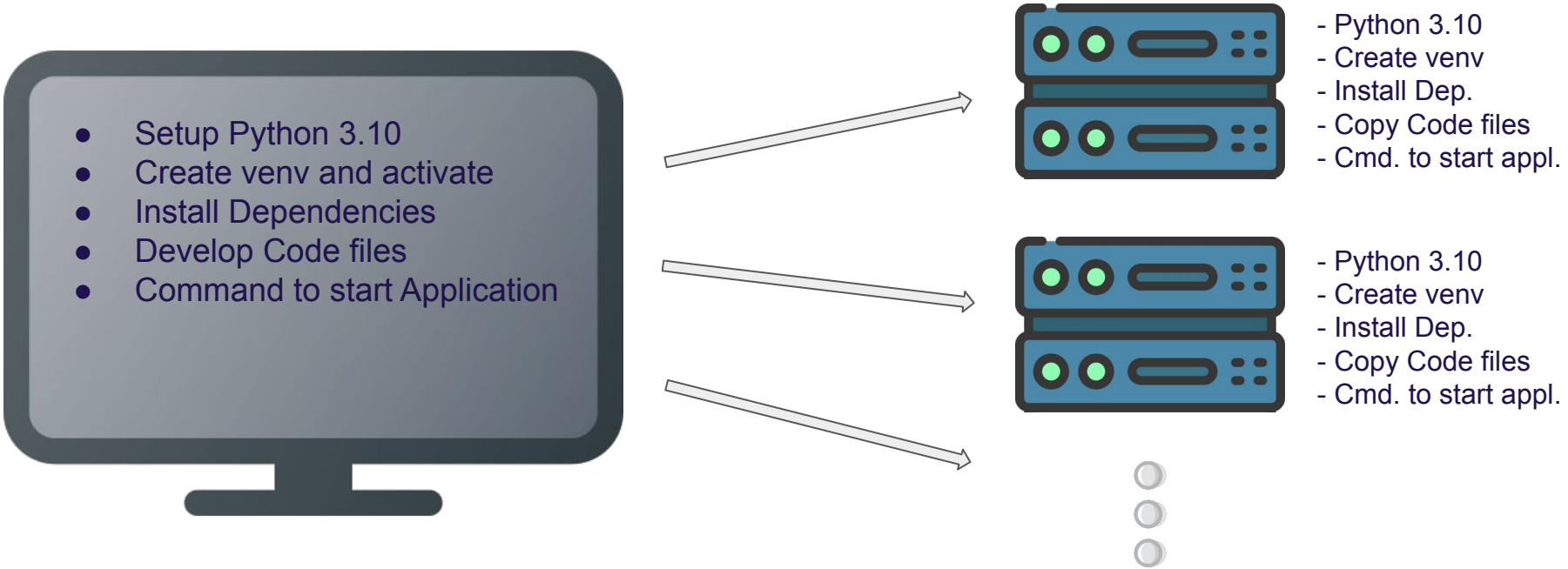
for Containerizing Application

Objectives

1. How to Run your Application on a Server? (Challenges in it)
2. What is meant by “Containerize your Application”
3. How to Containerize Application using “Docker”
4. How to create a “Docker Image”?
5. Basic Docker Commands

How to Run your Application on a Server?

- First, Build your application on your system, then take it to server



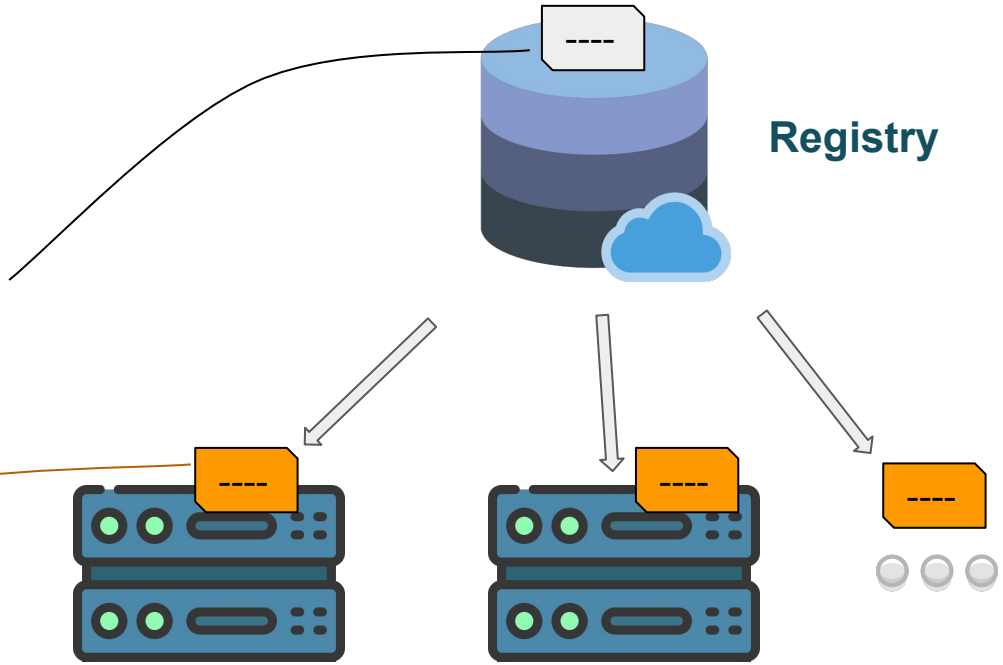
“ Process becomes Troublesome ”

Containerize your Application

- Once the application is build on your system, create a **template** of it
- Use **an instance** of that **template** to run the app anywhere

- Python:3.10 Installed
- Dependencies Installed
- Code files present
- Commands configured to start the appl.

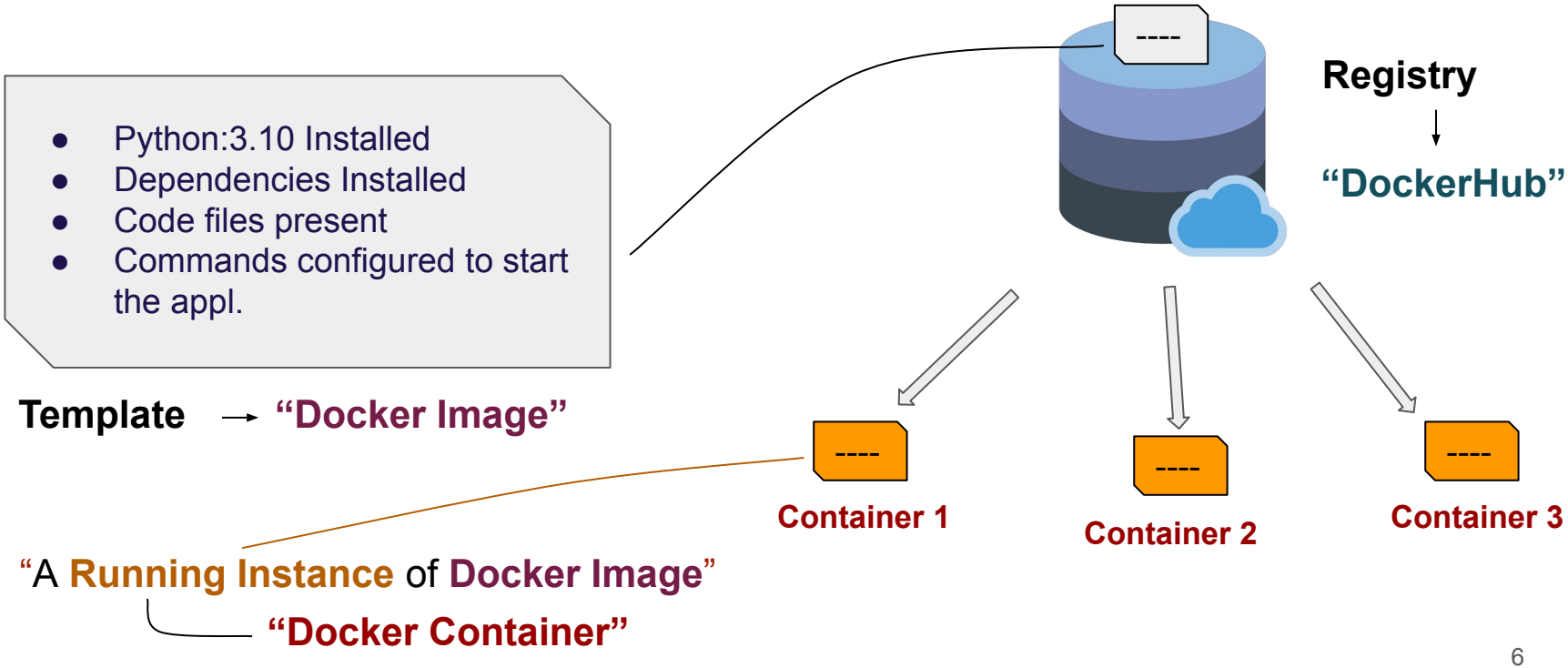
Template



"A **Running Instance** of **template**"

Containerize Application - “Docker”

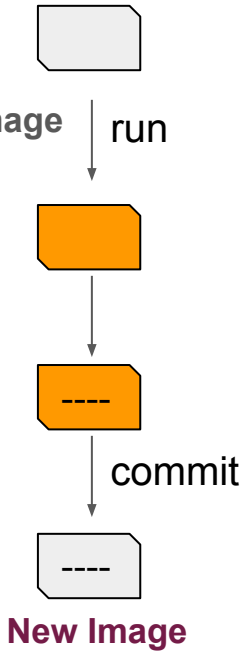
- Few terminologies to note:



How to create a “Docker Image”?

- Without **Dockerfile**

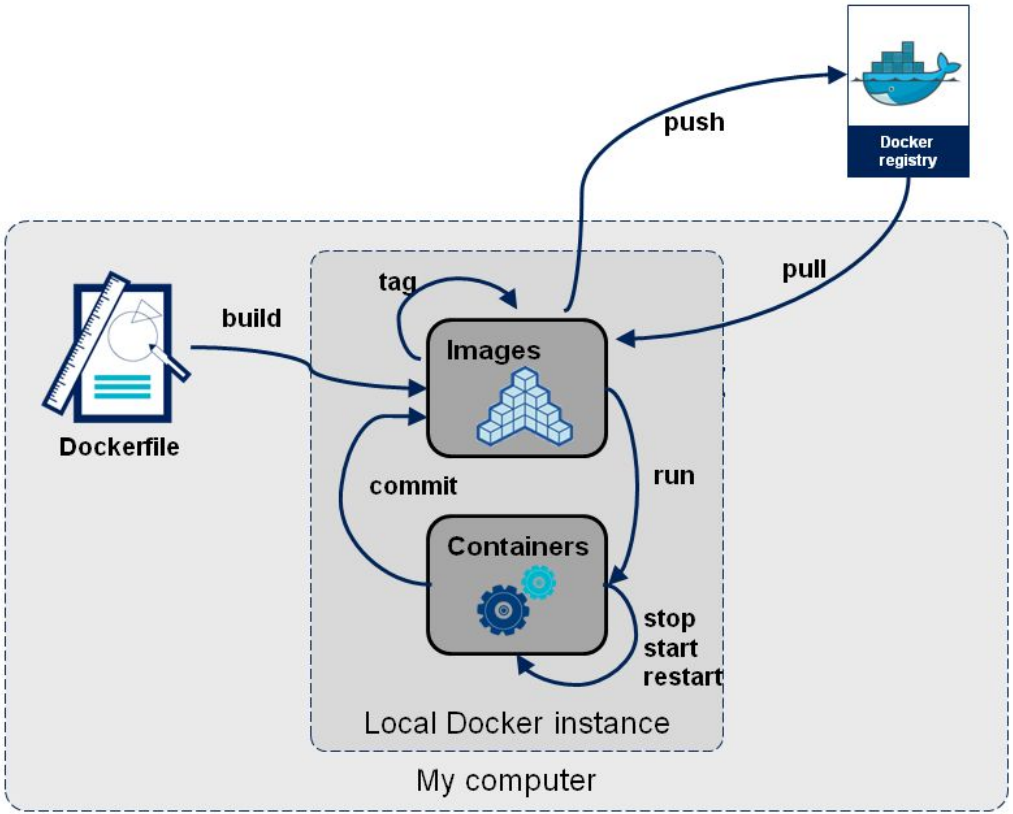
1. Run a Container using a **Base Image**
2. Inside Container
   a. **Manually** add your files
   b. Install dependencies
3. **Commit** the changes



- With **Dockerfile**

```
# pull python base image
FROM python:3.10
# copy application files
ADD . .
# specify working directory
WORKDIR /titanic_model_api
# update pip
RUN pip install --upgrade pip
# install dependencies
RUN pip install -r requirements.txt
# expose port for application
EXPOSE 8001
# start fastapi application
CMD ["python", "app/main.py"]
```

Basic Docker Commands




```
>> docker build . -t imgName:1
>> docker run --name=myApp imgName:1
        >> docker stop myApp
        >> docker start myApp
        >> docker restart myApp
        >> docker commit myApp newImg:1
>> docker tag newImg:1 username/newImg:2
>> docker push username/newImg:2
>> docker pull username/newImg:2
```

CICD

using GitHub Actions

Modularized application code

1. Training

```
>> pip install -r requirements/requirements.txt
>> python titanic_model/train_pipeline.py
```

2. Testing

```
>> pip install -r requirements/test_requirements.txt
>> pytest
```

3. Packaging

```
>> python -m build
```

4. Web framework - FastAPI

```
>> cd titanic_model_api
>> pip install -r requirements.txt
>> python app/main.py
```

5. Containerization - Docker

```
>> docker build . -t yrajm1997/titanic-api
>> docker push yrajm1997/titanic-api
```

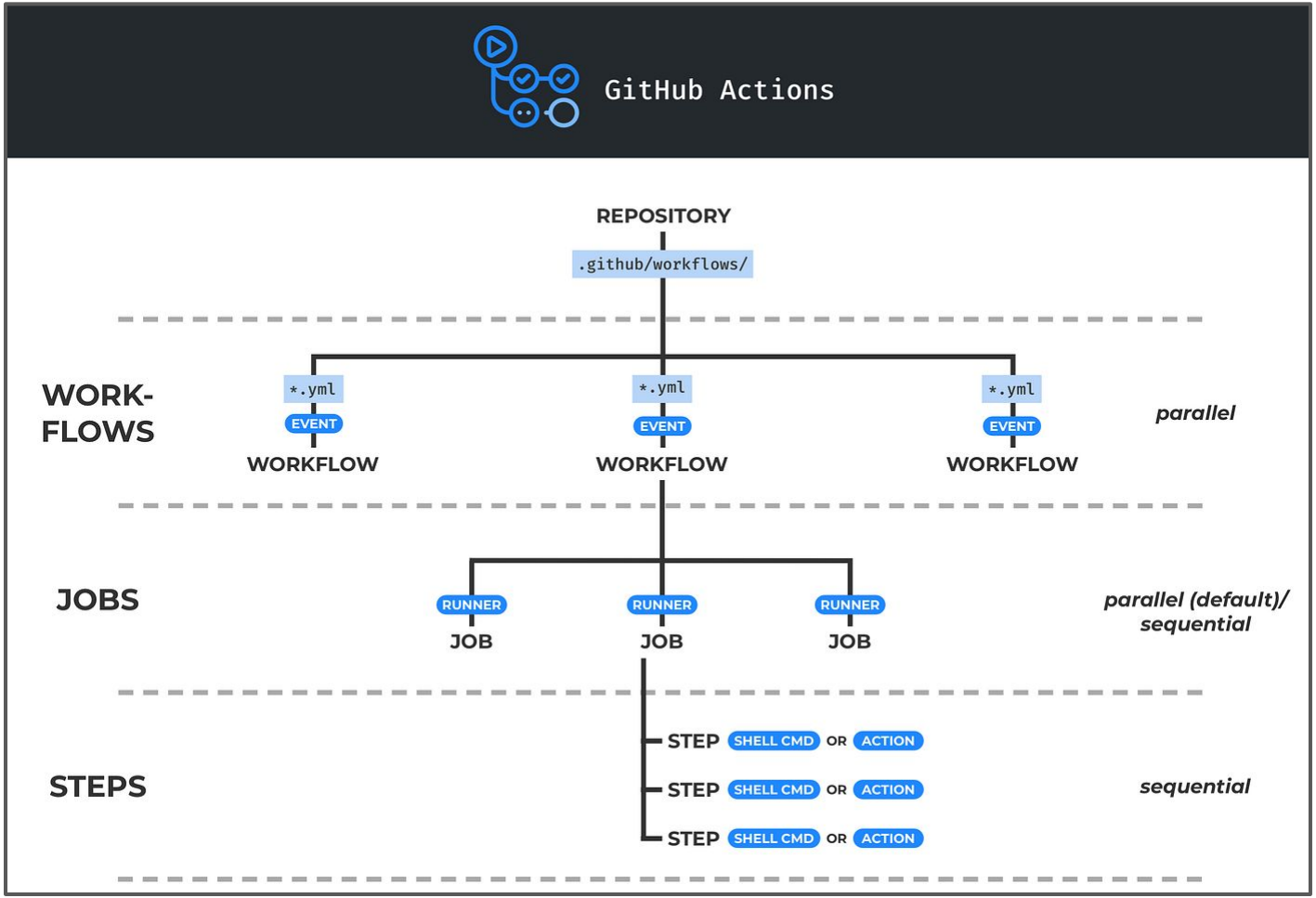
Setup CICD pipeline using GitHub Actions

CI:

- Create a repository on GitHub
- Add project files
- Add Docker credentials as Secrets in repository
- Create CI workflow and run

CD:

- Create an EC2 instance
- Install Docker on EC2 instance
- Add EC2 instance as a Self-hosted GitHub Runner
- Create CD workflow and trigger it
- Access App running on the public IP of the EC2 instance


GitHub Actions
REPOSITORY
.github/workflows/
WORK-FLOWS
*.yml
EVENT
WORKFLOW
*.yml
EVENT
WORKFLOW
*.yml
EVENT
WORKFLOW
parallel
JOBS
RUNNER
JOB
RUNNER
JOB
RUNNER
JOB
parallel (default)/ sequential
STEPS
STEP SHELL CMD OR ACTION
STEP SHELL CMD OR ACTION
STEP SHELL CMD OR ACTION
sequential

Thank you!