Empirical Study in Software Engineering

Welcome! Thank you for your availability to participate in this study. You will be asked to perform a set of orchestration tasks using Docker and Docker Compose technologies. The experiment should take between 50 minutes to 1 hour and 30 minutes in total.

*Obrigatório

Background questionnaire

Estimated time: 5 *mins*

Please answer the following questions about your current experience.

1. I consider myself experienced with visual programming tools. *

Marcar apenas uma oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Stronly Agree

2. What visual programming tools have you used in the past?

Marcar tudo o que for aplicável.

- ☐ Node-RED
- ☐ Blender Nodes
- ☐ Unreal Engine Blueprints
- ☐ Scratch
- ☐ Simulink
- ☐ Excel

Outra: ☐ ____________________

3. I consider myself experienced with orchestration frameworks. *

Marcar apenas uma oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Stronly Agree

4. What orchestration frameworks have you used in the past?

Marcar tudo o que for aplicável.

- ☐ Ansible
- ☐ Chef
- ☐ Puppet
- ☐ Salt
- ☐ Kubernetes

Outra: ☐ ____________________

5. I consider myself experienced with...

Marcar apenas uma oval por linha.

	Strongly Disagree	Disagree	Neutral	Agree	Stronly Agree
...the Linux operating system.	◯	◯	◯	◯	◯
...Docker.	◯	◯	◯	◯	◯
...Docker Compose for development purposes.	◯	◯	◯	◯	◯
...Docker Compose in production environments.	◯	◯	◯	◯	◯

Until now, approximately in how many projects have you ...

6. ...worked on which have used Docker Compose? *

Marcar apenas uma oval.

0	1	2	3	4	5	6	7	8	9	10	
◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	>10

7. ... created/updated a docker-compose.yml file? *

Marcar apenas uma oval.

	0	1	2	3	4	5	6	7	8	9	10	
	○	○	○	○	○	○	○	○	○	○	○	>10

8. ... used docker-compose.yml files created by others (colleagues or third parties)? *

Marcar apenas uma oval.

	0	1	2	3	4	5	6	7	8	9	10	
	○	○	○	○	○	○	○	○	○	○	○	>10

9. In the docker-compose files I've written, I've configured...

Marcar tudo o que for aplicável.

- [ ] ...volumes
- [ ] ...networks
- [ ] ...configs
- [ ] ...secrets

10. What software have you used to manage Docker or Docker Compose resources?

Marcar tudo o que for aplicável.

- [ ] Portainer
- [ ] Kitematic
- [ ] Admiral
- [ ] Dockstation

Outra: [ ] ____________________

General instructions

Read the following instructions very carefully.

During the experiment you should use the following resources:

All the required material (code and other files) you will need can be accessed from the root directory located at *~/Desktop/materials*. **You should not access any other content besides what is included in this directory**. Moreover, **you must only access the materials in the root directory once prompted to**.

You will also have access to the following tools:

- Firefox to access the internet.
- The system's default shell to execute necessary commands.
- Your preferred editor (installed in the machine) to access the contents of the root directory.
- The **Docker Composer** app to access, manipulate and manage Docker Compose stacks.

You can find the executable for Docker Composer (named Docker-Composer-1.0.0.AppImage) in the desktop. The app is already running but if for any reason you have to (re)start it, double-click on the executable.

Once you're ready to start, **advance to the next section**.

Tutorial on Docker Composer

Estimated time: 10 *mins*

Start by following along this tutorial to familiarize yourself with the **Docker Composer** tool. The overall purpose of Docker Composer is to provide a visual alternative to edit and visualize stacks defined in Docker Compose. You can find an in-depth guide detailing how to use the tool as well as other useful documentation in the wiki available at:

- https://github.com/Kubix20/docker-composer/wiki

Start by carefully reading the guide to learn the basics of the tool, paying close attention to the introduction and "How to use" sections. You can always refer to the wiki when in doubt.

The example in this tutorial defines a service stack, comprising a **web app** and a **redis database**. The web app service just outputs whether it has successfully connected to the database or not. To check the connection status, you can send a GET request to the root (/) endpoint at port 80. Both services should execute locally (i.e. **not** Swarm) meaning that each service runs in a single container (i.e. two containers in total).

In summary, the resulting stack should include:
- A **web** service (web - kubix20/webapp_redis:latest - 80).
- A **redis** database (redis - redis:alpine - 6379).

Note: The details specified above are in the format (*key* - *image* - *exposed ports*).

The resulting stack should also consider the following:

- Services must set the keys and images as specified above.
- The web app container should start after the database container.
- A volume named storage to persist the redis data.

Expected behaviour: The web app successfully connects to the database.

11. Follow along these next steps. Tick each step as you complete it.

Marcar tudo o que for aplicável.

☐ 1. Open the Docker Composer tool (already running on the machine).

☐ 2. Add a new service by right clicking in the graph area and selecting ‘New service’ in the context menu. This will be the **web** service.

☐ 3. Set the key field to web and image to kubix20/webapp_redis either in the artifact or in the properties editor.

☐ 4.Open the Docker Hub page for the image kubix20/webapp_redis by clicking on the icon above the image input on the properties editor to learn more about the service.

☐ 5. Add a port entry in the properties editor mapping port 80 on the container to 4000 on the host.

☐ 6. Using the image palette search for the official redis image and drag it onto the graph area. This will be the **redis** service.

☐ 7. Set the key to redis and image tag to alpine.

☐ 8. Add a depends_on dependency from the web service to the redis service by dragging an arrow from the depends_on anchor point on the web service.

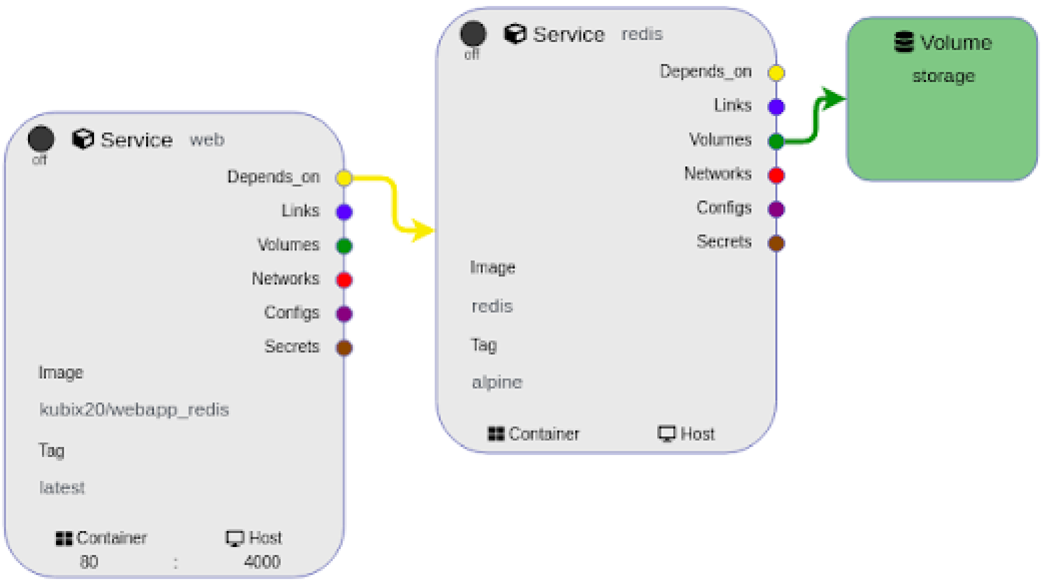
☐ 9. Add the environment variable in the properties editor REDIS_HOST=redis in the web service.

☐ 10. Add a volume via the context menu and set the key to storage.

☐ 11. Connect the redis service to the volume through the corresponding volume anchor point (green).

☐ 12. Select the connection and set the target to /data in properties editor. This is the path where the volume will be mounted in container.

The stack should now look like this:



Next, it's time to test the stack.

12. Follow along these next steps. Tick each step as you complete it.

Marcar tudo o que for aplicável.

- ☐ 13. Click on the Start button in the toolbar to run the app.
- ☐ 14. Check the output in the terminal and verify that the message “Connected to DB” is printed by the web container, indicating that the web service has successfully connected to the database.
- ☐ 15. Make a GET request to localhost:4000 (using your preferred method, e.g. curl or browser) and verify the response “Connected to db” is received.
- ☐ 16. Click on the Stop button to stop the running stack.
- ☐ 17. Export the stack and save it in the tutorial folder located in the root directory with the name *tutorial.yml*.

If you've ticked all the boxes, advance to the next section.

Tasks

Estimated time: 30 *mins*

Read the following instructions very carefully.

You will be asked to perform a set of orchestration tasks with Docker Compose.

In the root directory, you will find folders containing the material for each task named t*, where * is the task number (e.g. t1 for task 1). **For each task**, **you must only access the contents of the corresponding folder while performing said task**. Moreover, **you must execute tasks** (**and subtasks**) **sequentially**, meaning that you cannot change your answers for previous tasks.

All tasks are preceded by a section labeled T* - Setup, where * is the task number (e.g. T1 - Setup), containing instructions you must follow before advancing to the actual task. These ensure that you abide by the guidelines described in this section.

When starting a task, carefully read the description once, in full. Once you’ve read the description and before you start solving the task, register the current time on the input labeled Start time. Likewise, once you finish the task register the current time on the input labeled Finish time. You can find both inputs below the description of the task.

Some tasks require the creation/edition of a Docker Compose stack. Keep in mind that the focus of the exercise is on the orchestration process and not in the development of the components that are part of the stack. In this sense, the requirements must be satisfied **strictly through the edition of the stack**.

Once you’re ready to start, **advance to the next section**.

T1 - Setup

Before proceeding, follow the next steps:

1. Navigate to the folder named **t1** from the root directory. You must exclusively access the contents of this folder while performing this task.
2. Load the stack in this folder with Docker Composer by clicking on the 'Open' button and selecting the folder t1.

Once you're ready to start, **advance to the next section**.

T1 - Analyzing a stack

In this task you will analyze a Docker Compose stack for a **voting app**.

Begin by carefully examining the contents of the stack. You can also start the app and access the services through their exposed ports to visualize it in action.

Important: You can learn more about each service in the corresponding image page on Docker Hub.

When you're ready, answer the following questions.

13. Start time *

____

Exemplo: 08:30

Exercise 1

14. Answer true or false to the following statements: *

Marcar apenas uma oval por linha.

	True	False
Some services use the default network.	⬭	⬭
The votes are stored in the redis service.	⬭	⬭
The named volume db-data is used to provide configurations to the postgres service at runtime.	⬭	⬭
The redis service always exposes port 6379 on the host.	⬭	⬭
The vote service uses a locally built image.	⬭	⬭

Exercise 2

15. What services depend on the redis service? (Answer in the format [services], e.g. ser1, ser2,...) *

16. What ports are exposed to the host by which services? (Answer in the format: service-[ports], e.g. container-123,124) *

17. What networks are used and what services are attached to each one? (Answer in the format: network-[services], e.g. net1-ser1, ser2,...; net2-ser1) *

18. Finish time *

Exemplo: 08:30

T2 - Setup

Before proceeding, follow the next steps:

1. Confirm that the stack is properly stopped.
2. Close all resources from previously used folders.
3. Navigate to the folder named **t2** from the root directory. You must exclusively access the contents of this folder while performing this task.
4. Load the stack in this folder with Docker Composer.

Once you're ready to start, **advance to the next section**.

T2 - Fixing a stack

Consider the stack for a simple app to register and view TODO notes.

The stack architecture is as follows:
- A mongoDB database to store the TODOs (mongo:4.2.0 - 27017)
- A backend server to expose an API to... (kubix20/todoapp_server - 3000)
- A client frontend for viewing and registering TODOs (kubix20/todoapp_client - 3000)

Note: The details specified above are in the format (*image - exposed ports*).

The backend server looks for the mongoDB service running on the host **mongo** and port 27017 without authentication required.

The frontend client proxies all API requests to the server service which is expected to be running on the host **server** and port 3000.

Unfortunately, the app isn't currently working as expected as users are unable to register new TODOs. Locate and fix the bugs so that the app behaves as expected.

Important: You must achieve the expected behaviour **without adding or removing any of the existing artifacts** (services, networks and volumes).

Note: The issue is unrelated to the stdin_open property on the client. It must be set to true for the client to execute properly.

Hint: Run the stack to observe the faulty behavior.

Important: Once you have finished the task, export the stack and save it in the t2 folder with the name docker-compose.yml.

19. Start time *

____

Exemplo: 08:30

20. Finish time *

____

Exemplo: 08:30

T3.1 - Setup

Before proceeding, follow the next steps:

1. Confirm that the stack is properly stopped.
2. Close all resources from previously used folders.
3. Navigate to the directory named **t3.1** from the root directory. You must exclusively access the contents of this folder while performing this task.
4. Load the stack in this folder with Docker Composer. Once loaded, the graph area should be empty.

Once you're ready to start, **advance to the next section**.

T3.1 - Creating a stack

Create a stack comprised of a **web app** and a **postgres database**. The web app outputs whether it has connected to the database or not. To check the connection status, you can send a GET request to the root endpoint (/) at port 80.

In summary, the stack should include:

- A **web app** service (web - kubix20/webapp_postgres:latest - 80)
- A **postgres database** (db - postgres:9.4 - 5432)

Note: The details specified above are in the format (*key* - *image* - *exposed ports*).

The web service accepts the following environment variables:

- DB_USER (default value ""), to specify the user when connecting to the postgres database.
- DB_PASSWORD (default value ""), to specify the password when connecting to the postgres database.

You must use the configuration files provided in the folder **t3.1**. More specifically:

- /postgres contains an environment file named credentials with variables to set the credentials in the **postgres database**. Note that you must set these variables by referencing the file and **not** by setting the individual environment variables within it.

The resulting stack should also consider the following:

- Services should set the keys and images as specified above.
- The web service should be exposed to the host on port 4000.
- The web container should start **after** the database container.
- A named volume called db-data to persist the database data mounted at /var/lib/postgresql/data.
- A custom network named my-net to which both services should be attached.

Expected behaviour: The web app successfully connects to the database.

The task is successfully complete **only if** the expected behavior is achieved **and** all other requirements satisfied.

Important: Once you have finished the task, export the stack and save it in the t3.1 folder with the name docker-compose.yml.

21. Start time *

Exemplo: 08:30

22. Finish time *

Exemplo: 08:30

T3.2 - Setup

Before proceeding, follow the next steps:

1. Confirm that the stack is properly stopped.
2. Close all resources from previously used folders.
3. Navigate to the folder named **t3.2** from the root directory. You must exclusively access the contents of this folder while performing this task.
4. Load the stack in this folder with Docker Composer.

Once you're ready to start, **advance to the next section**.

T3.2 - With secrets

Alter the stack (as defined in T3.1) so that the database credentials are provided to the **web** service through secrets instead of environment variables.

You must use the configuration files provided in the folder **t3.2**. More specifically:

- /postgres/secrets contains two files (user and password) with matching credentials to the ones in the credentials file (used in the db service). The content of these files should be used as secrets, named db_user and db_password respectively.

Expected behaviour: The web app successfully connects to the database.

The task is successfully complete **only if** the expected behavior is achieved **and** all other requirements satisfied.

Note: The results service expects the credentials as secrets **or** environment variables. This means that the app will work as expected if the environment variables are correctly set, even if the secrets are not. Ensure that the environment variables are not set in the **web** service to test the stack.

Important: Once you have finished the task, export the stack and save it in the t3.2 folder with the name docker-compose.yml.

23. Start time *

Exemplo: 08:30

24. Finish time *

Exemplo: 08:30

Post-experiment questionnaire

Estimated time: 8 *mins*

You've completed all the tasks and have reached the last step of the experiment.

Please answer the following questions in regards to your experience.

25. Mark the answers that best reflect your opinions. *

Marcar apenas uma oval por linha.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
It was easy working in the remote machine.	◯	◯	◯	◯	◯
The environment was distracting.	◯	◯	◯	◯	◯
I found the procedure instructions complex and difficult to follow.	◯	◯	◯	◯	◯
I found the task descriptions complex and difficult to follow.	◯	◯	◯	◯	◯
Overall, I found the tool easy to learn.	◯	◯	◯	◯	◯
Overall, I found the tool difficult to use.	◯	◯	◯	◯	◯
I found it difficult to understand stacks with the tool.	◯	◯	◯	◯	◯
I found it easy to define stacks with the tool.	◯	◯	◯	◯	◯

Mark the answers that best reflect your opinions.

26. I find the visual map of artifacts.... *

Marcar apenas uma oval por linha.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
...helpful to understand stacks with less effort.	⬭	⬭	⬭	⬭	⬭
...helpful to understand stacks more quickly.	⬭	⬭	⬭	⬭	⬭
...helpful to define stacks with less effort.	⬭	⬭	⬭	⬭	⬭
...helpful to define stacks more quickly.	⬭	⬭	⬭	⬭	⬭

27. I find the integration with Docker Hub (image palette and links to the docs)... *

Marcar apenas uma oval por linha.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
...helpful to understand stacks with less effort.	⬭	⬭	⬭	⬭	⬭
...helpful to understand stacks more quickly.	⬭	⬭	⬭	⬭	⬭
...helpful to define stacks with less effort.	⬭	⬭	⬭	⬭	⬭
...helpful to define stacks more quickly.	⬭	⬭	⬭	⬭	⬭

28. I find the visual feedback of running stacks (service and stack LEDs)... *

Marcar apenas uma oval por linha.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
...helpful to understand the state of a stack with less effort.	⬭	⬭	⬭	⬭	⬭
...helpful to understand the state of a stacks more quickly.	⬭	⬭	⬭	⬭	⬭

29. I find executing commands in the UI (start and stop)... *

Marcar apenas uma oval por linha.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
...helpful to define stacks with less effort.	⬭	⬭	⬭	⬭	⬭
...helpful to define stacks more quickly.	⬭	⬭	⬭	⬭	⬭

30. In comparison to the conventional procedure (editing a docker-compose.yml file and docker cli)... *

Marcar apenas uma oval por linha.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
...I believe this tool would reduce the effort required to define Docker Compose stacks.	⬭	⬭	⬭	⬭	⬭
...overall, I found the tool useful.	⬭	⬭	⬭	⬭	⬭
...a Docker Compose stack visualized with the tool would be more difficult to understand.	⬭	⬭	⬭	⬭	⬭
...overall, I think this tool does not provide an effective solution to define Docker Compose stacks.	⬭	⬭	⬭	⬭	⬭
...overall, I think this tool makes an improvement to the stack definition process.	⬭	⬭	⬭	⬭	⬭

31. Mark the answers that best reflect your opinions *

Marcar apenas uma oval por linha.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
This tool would make it easier for practitioners to define Docker Compose stacks.	⬭	⬭	⬭	⬭	⬭
Using this tool would make it easier to communicate the stack architecture to other practitioners.	⬭	⬭	⬭	⬭	⬭
I would recommend this tool to work with Docker Compose.	⬭	⬭	⬭	⬭	⬭
I would like to use this tool in the future.	⬭	⬭	⬭	⬭	⬭
It would be easy for me to become skillful in using this tool to work with Docker Compose.	⬭	⬭	⬭	⬭	⬭

32. What do you think can be improved upon in the tool?

33. Any comments? (about your experience, the experiment process, the tool itself, ...)

End

Congratulations! You have completed the experiment.

Thank you for your participation!