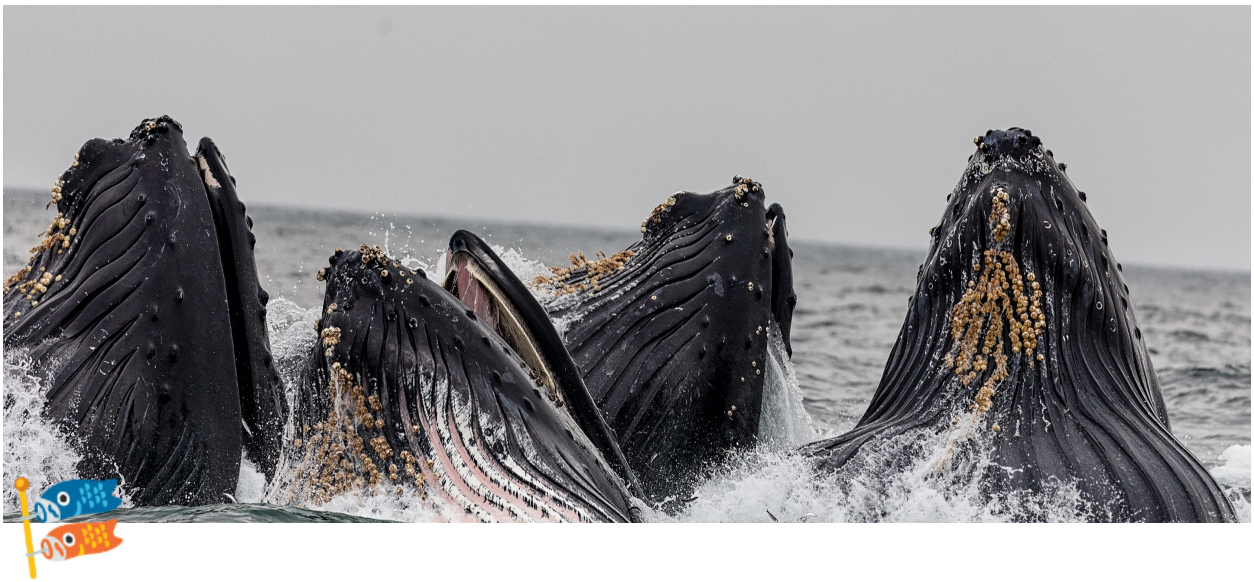
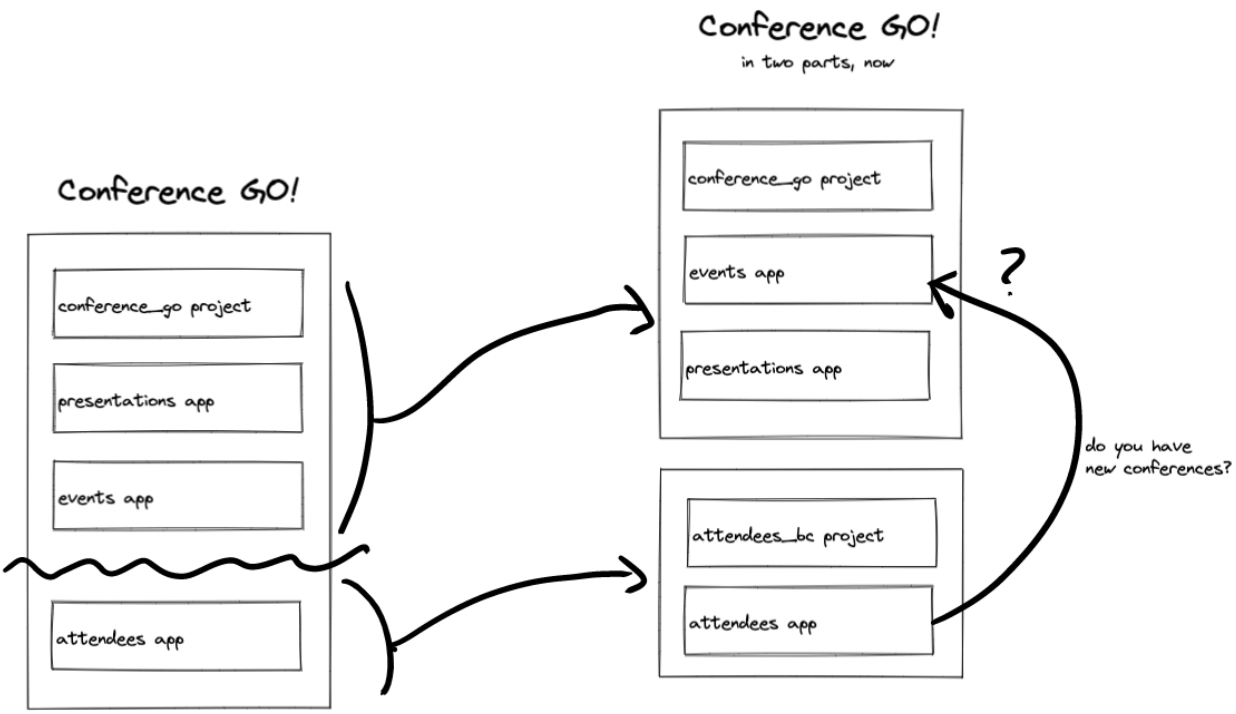


Docker Networks & Microservices Architecture — Q&A

🔧 **What you're building:**

- Splitting up your Conference GO app into multiple microservices
- …which communicate with each other over a shared Docker bridge network
- …using a **VO** to cache data that is transferred from one microservice to another



Bridge Network

A bridge network uses a “software bridge” which allows containers connected to the same network to communicate and share files with each other, while providing isolation from containers which are not connected to that bridge network.

When you start a container without specifying a specific network to join, the bridge network named "bridge" is the default network that Docker connects this unspecified container to.

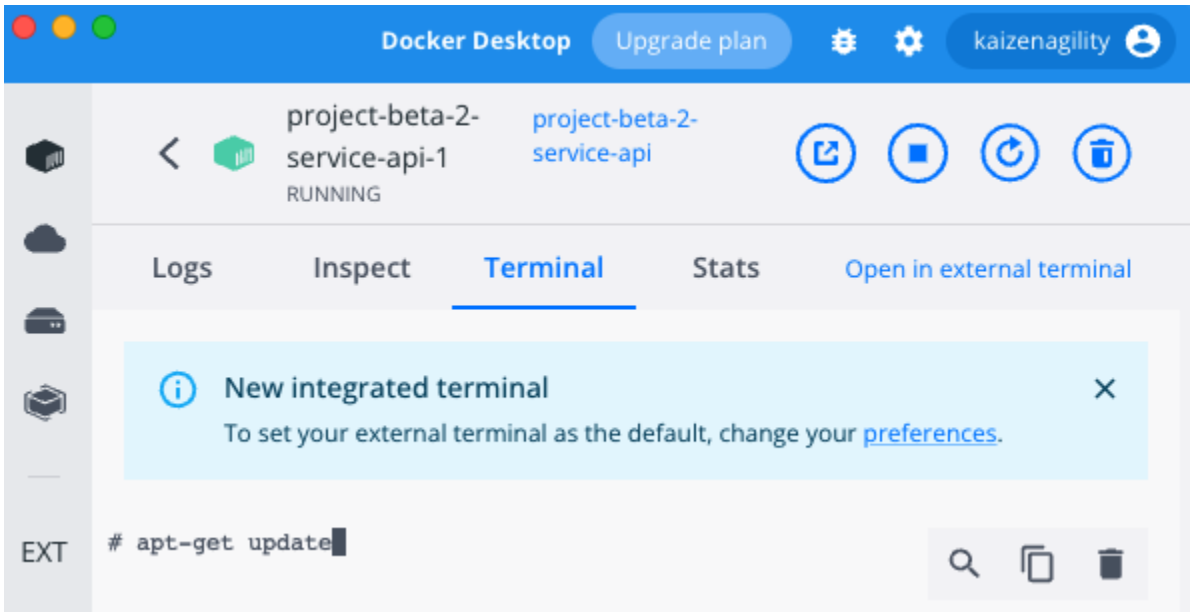
LAB COMMANDS SIMPLIFIED:

```
# create a docker network
docker network create --driver bridge <network_name>

# start containers within a network
docker run -dit --name <container_1> --network <network_name> python:3 bash
docker run -dit --name <container_2> --network <network_name> python:3 bash
```

```
# list the networks and containers you have started in Docker
docker network list
docker container list

# inspect the network for more details, such as IPv4 addresses of containers
docker network inspect <network_name>

# go inside a docker container, and use the terminal inside that container
docker attach <container_1>

# install ping package, so that container can send messages to the outside world
apt-get update
apt-get install iputils-ping -y

# ping something with four packets! ("-c" specifies number of packets to send)
ping -c 4 <IP address of another container>

# exit the terminal inside of that container
exit

# stopping and deleting containers when you're done
docker container stop <container_1> <container_2>
docker container rm <container_1> <container_2>
```

NOTE:

Running `Docker attach` in Terminal is the same as doing this in your Docker Desktop app:



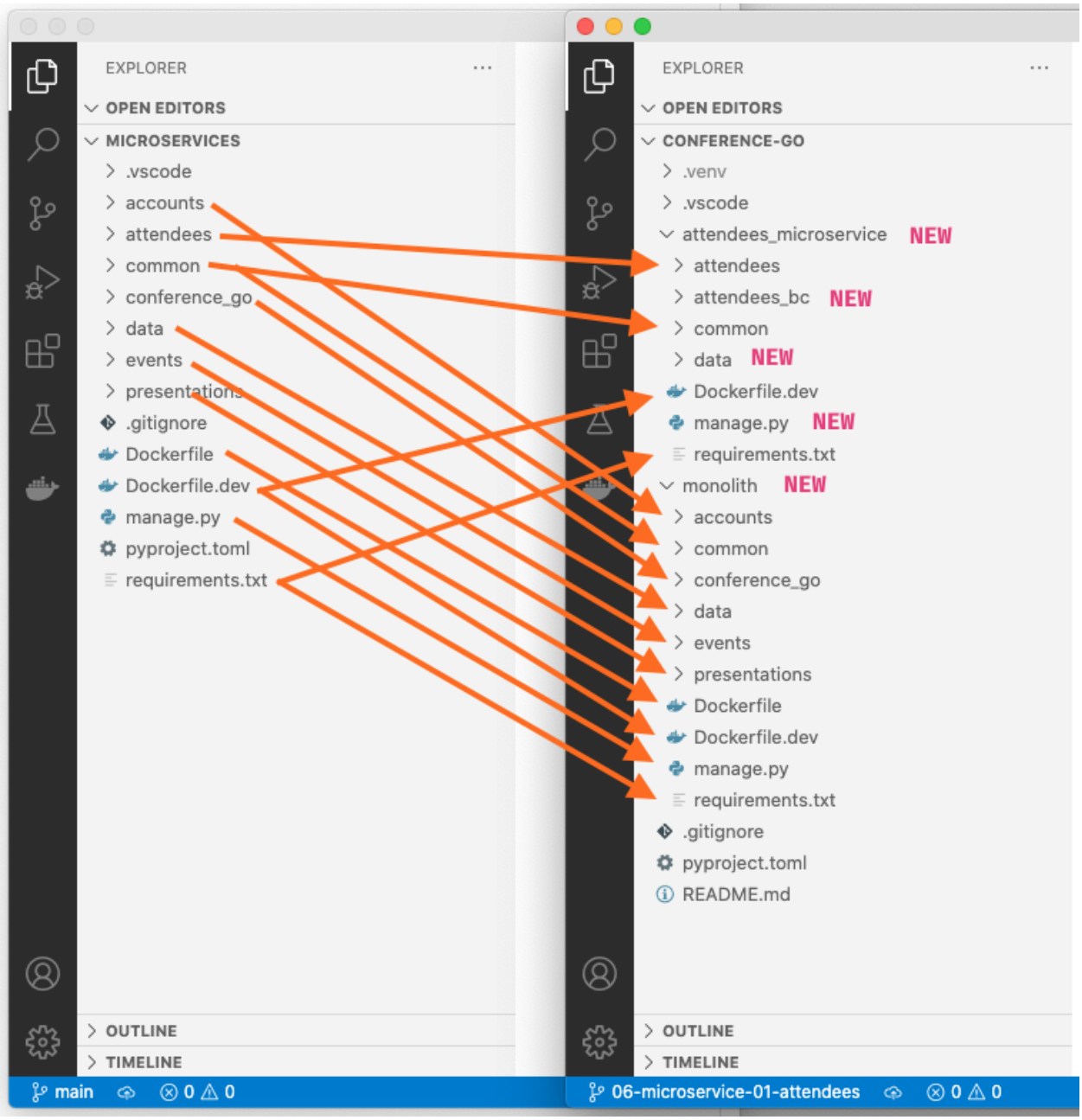
Click on the container you want to run commands from in your container tab in Docker Desktop. Then go to the Terminal tab, or “Open in external terminal.”

Monolith to Microservices

In your lab, you’ll be moving folders out of the “Monolith” application into separate microservices, starting with the “Attendee” microservice.

- Each microservice is its own separate Django project, with its own Django file structure (manage.py, settings, Dockerfiles, etc.)
- Each microservice sits in its own Docker container.
- Each container can only talk to other microservices across the bridge network using a poller. (See the poller.py script here.)

Your new folder structure:



MICROSERVICES ARCHITECTURE

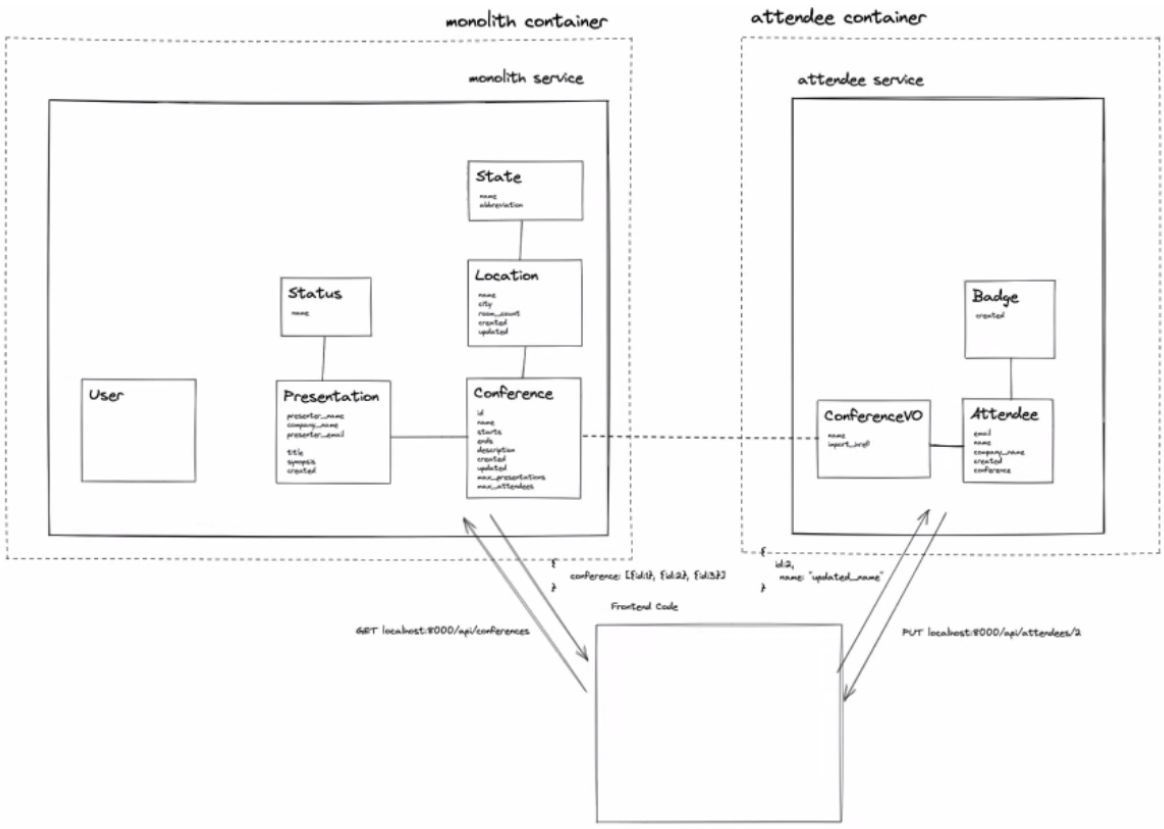


Diagram by Warren Longmire in this morning's lecture.

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/6870134c-92a2-43c8-a763-6441c1cbbd6a/image.png

Why use a VO?

- Saves time: http calls are slower and are costly
- Immediately accessible information, cached from the other microservice

```
# In "attendees_microservice/attendees/models.py":
# Replace the Conference class in a monolith with a ConferenceVO class
# This VO pulls data from another microservice
```

```
class ConferenceVO(models.Model):
    import_href = models.CharField(max_length=200, unique=True)
    name = models.CharField(max_length=200)


# In "attendees_microservice/attendees/api_views.py":
# Create a new encoder with model: ConferenceVO
# Replace all previous encoders with this VO encoder

class ConferenceVODetailEncoder(ModelEncoder):
    model = ConferenceVO
    properties = ["name", "import_href"]
```

*Actually, in the real world, you're not likely to use a VO and a **Cron** job (for task scheduling) to do this. You're more likely to cache web requests using an in-memory data structure store like **Redis**.*

How does Cron task scheduling work?

https://youtu.be/v952m13p-b4

Adding the poller script as a cron job

In the *attendees_microservice/attendees_bc/settings.py*, add the following code:

```
INSTALLED_APPS = [
    "django_crontab",
    ....
]

CRONJOBS = [
    ("* * * * *", "attendees.poll.get_conferences"),
]
```

Got questions:

Add as comments below!