Deploy Like a Boss.

Evgeny Shulman
CTO at databand.ai
About Our Speaker: Evgeny Shulman

Evgeny is Co-Founder & CTO at Databand.ai.

- He’s been building pipelines since 2004 at Intel, adTech, and Oracle Data Cloud.

- At Databand.ai, he helps data engineering teams ensure reliable delivery of quality data with a purpose-built monitoring system.

- Huge fan of Airflow (scheduling performance improvements, AIP-31, DebugExecutor)
Process Quality

- **Job Performance**
  Are queries and jobs running efficiently?

- **Pipeline Latency**
  Are pipelines running on time?

- **Pipeline Execution**
  Is data flowing?

Data Quality

- **Data Content**
  Are there significant changes or issues within the data?

- **Data Structure**
  Is the data shape valid and complete?

- **Data Freshness**
  Is data arriving on time?
Databand
Data Pipeline Reliability
Managing Airflow in complex environments is tough.

How to succeed?

- Increase Apache Airflow deployment robustness, use the right Platform for that. - We have only 20 minutes today!
- Friendly development environment.
After attending the session, Airflow engineers will:

- Know the differences between Airflow Kubernetes and Local Executor
- Understand the advantages of some kinds of deployments
- Deploy how to incorporate all kinds of deployments for their day-to-day needs
“One time I tried to explain Kubernetes to somebody. When we both didn’t understand it.”

SwiftOnSecurity (twitter)
Making my mind around Airflow Deployment

- “We use Kubernetes to run everything”
- “Airflow Kubernetes integration is great”
- “If you need scale you should use Kubernetes”

The decision is made
If everything works.. I am the BOSS . What happens if I am not?

As a data engineer I am not k8s specialist.

- `Kind` didn’t work for me
- Deployment is stacked
- Some secrets are not defined
- It took me 20 minutes to understand that I didn’t publish my image.

Can Google Search be wrong?
I should wait another 200ms before making decision
What’s wrong with saying

“I use k8s at my Airflow cluster”.

➔ Python Operator with Kubernetes submission code
➔ Kubernetes Operator
➔ Kubernetes Executor
➔ Helm Deployment
What’s wrong with saying “I use k8s at my Airflow cluster”.

- Python Operator with Kubernetes submission code -> Runs somewhere, submit USER CODE to K8s
- Kubernetes Operator -> Runs somewhere, submit USER CODE to K8s
- Kubernetes Executor -> Runs somewhere, submit Airflow TASKS to k8s
- Helm Deployment -> Scheduler runs on k8s, submit TASKS somewhere
- DEPLOYMENT X EXECUTOR X OPERATOR
Scoping is everything

- I like to be in control of my code.
- I like to be a boss of my deployment
This is what made us(me) use docker in our dev environment.

As a software developer I test my web service locally.

It’s still great. Simplicity!


**Airflow on docker-compose**

- DAG mounts
- Searchable LOGs
- Easy to change config
- Easy to redeploy docker (full control)
Why and When?

My Use cases

● I want to update my Airflow docker image
  ○ optimize docker build, add python dependencies, add my company libraries)
  ○ Docker build .. && docker-compose up

● I want to debug Airflow command, but I don't have DB, UI and scheduler
  ○ docker-compose up

● I want to test another integration (like EMR), but I don't know where to start..
  ○ docker-compose (AWS permissions on my machine, can easily change docker)
Local Development

➔ If possible, avoid airflow.cfg, use variables (env into docker-compose, helm and so on )

➔ Have a setup script, so you can run things from local/pycharm (AIRFLOW_HOME +
  AIRFLOW__CORE__SQL_ALCHEMY_CONN=postgresql+psycopg2://
  airflow:airflow@localhost:5494/airflow

➔ !! Mounts not always works (FORK)
**When I become a real boss of the deployment**

- Local Python and Docker-compose - for fast, iterative development
- Production: Kubernetes with Helm Chart/...
Challenges we had to solve

- Dag deployment, credentials, many others
- Managing connections and variables (HELM charts has their own systems)

-> Use external provisioning script (with airflow command)
-> Use CI/CD variables to inject values, you can run that docker just as a simple CI/CD job
-> You can run that script locally for docker-compose as well as for helm for..
What worked for us with K8S

➔ Easier to run k8s jobs (via executor or operator)
➔ Hard to maintain version updates.
➔ Variables and Connections
➔ Managed Airflow. (not all of them k8s, or they might be, but you will never know)
Our Development Cycle:

➔ Starting with docker-compose
➔ Moving to production with Helm + Kubernetes
What worked for me

→ Simple is better than complex
→ I can create a lot of different environments
→ My co-workers are actually happy
Suggestions for Next Meetup Topics?
Thank you!

See you at our next meetup!