**Researching Airflow deployment**

The user asks for a step-by-step guide on setting up Airflow for production deployment on-premise. I’ll need to search for authoritative articles and tutorials related to Airflow. My plan is to search for relevant queries like "Apache Airflow production deployment on-premise" and review the official documentation. I also need to include a detailed breakdown with sections like prerequisites, security, and scaling, along with gathering citations from at least 5 domains. I’ll make sure to format the answer clearly using markdown.

**Planning Airflow deployment**

I need to search for "Apache Airflow production deployment on-premise step by step" and other related guides. The official docs will help clarify things like database backend, executors, and system configuration. I should include citations for each step. For the outline, I'll provide a summary paragraph first and then cover key sections: infrastructure planning, choosing an executor, setting up the database and message broker, installing prerequisites, configuring the Airflow service, security, monitoring, scaling, and fleet management.

**Formulating Airflow setup guide**

To set up Airflow for on-premise production, I’ll break it down into parts: prerequisites, installation (using pip or Docker Compose), database, and message broker setups. I'll go into configuring Airflow for production, handling high availability with CeleryExecutor, and ensuring security. Monitoring tools like Prometheus and Grafana will be important. Additionally, syncing DAGs, scaling, and backup/disaster recovery processes will be covered. I will include citations from various sources like official docs, medium articles, and other tech sites.

Thought for 12 seconds

Before diving into the details, here’s a high-level overview of the on-premise Airflow production deployment process:

1. **Plan the infrastructure**: define hardware sizing, networking, and high-availability requirements.
2. **Prepare core services**: set up a fault-tolerant external metadata database (PostgreSQL/MySQL) and a broker (RabbitMQ/Redis) if using CeleryExecutor.
3. **Install Airflow and dependencies**: create isolated Python environments, install Airflow with needed extras, and ensure all nodes share the same code and configurations.
4. **Configure Airflow**: update airflow.cfg for production settings—executor, parallelism, secrets backend, and logging.
5. **Set up high availability**: deploy multiple schedulers and workers; use a load balancer for the web server and a replication strategy for the database.
6. **Harden security**: enable RBAC, SSL/TLS, and secrets management.
7. **Implement CI/CD and DAG sync**: automate DAG deployment via Git, Ansible/Chef/Puppet, or shared storage.
8. **Enable monitoring and alerting**: integrate Prometheus, Grafana, and log aggregation (ELK/EFK).
9. **Plan for scaling, backups, and disaster recovery**.

**1. Prerequisites and Planning**

**1.1 Infrastructure Requirements**

* **Compute**: Separate VMs or bare-metal servers for the webserver, scheduler(s), and worker(s). Ensure they have CPU/RAM to match your DAG load.
* **Networking**: Private network with low latency between Airflow components. Open ports 5432 (PostgreSQL), 5672 (RabbitMQ), 8080 (Web UI).
* **Storage**: Shared or synchronized storage for DAGs, logs, and plugins across nodes ([Apache Airflow](https://airflow.apache.org/docs/apache-airflow/stable/administration-and-deployment/production-deployment.html?utm_source=chatgpt.com)).

**1.2 Airflow Components**

* **Metadata DB**: External PostgreSQL/MySQL—built-in SQLite is only for testing ([Apache Airflow](https://airflow.apache.org/docs/apache-airflow/stable/administration-and-deployment/production-deployment.html?utm_source=chatgpt.com)).
* **Executor**: For small teams, LocalExecutor can work; for scale & HA, use CeleryExecutor with RabbitMQ/Redis ([Apache Airflow](https://airflow.apache.org/docs/apache-airflow-providers-celery/stable/celery_executor.html?utm_source=chatgpt.com)).
* **Scheduler**: Run two or more schedulers in standby; only one actively scheduling to reduce single point of failure.
* **Webserver**: Front-end UI, typically behind a load balancer.

**2. Database and Message Broker Setup**

**2.1 PostgreSQL Configuration**

1. Install PostgreSQL on a dedicated server.
2. Create database and user:
3. CREATE DATABASE airflow;
4. CREATE USER airflow WITH PASSWORD 'strong\_password';
5. GRANT ALL PRIVILEGES ON DATABASE airflow TO airflow;
6. Enable logical replication for HA if needed.
7. In airflow.cfg, set:
8. [core]
9. sql\_alchemy\_conn = postgresql+psycopg2://airflow:strong\_password@db-server:5432/airflow
10. ``` :contentReference[oaicite:3]{index=3}.

**2.2 RabbitMQ/Redis for CeleryExecutor**

1. Install RabbitMQ or Redis:
2. sudo apt-get install rabbitmq-server
3. Create a vhost and user for Airflow:
4. rabbitmqctl add\_vhost /airflow
5. rabbitmqctl add\_user airflow strong\_password
6. rabbitmqctl set\_permissions -p /airflow airflow ".\*" ".\*" ".\*"
7. Configure in airflow.cfg:
8. [celery]
9. broker\_url = amqp://airflow:strong\_password@broker-server:5672/airflow
10. result\_backend = db+postgresql://airflow:strong\_password@db-server:5432/airflow
11. ``` :contentReference[oaicite:4]{index=4}.

**3. Airflow Installation**

**3.1 Python Environment**

1. Install system dependencies (e.g., gcc, libmysqlclient-dev, python3-venv).
2. Create and activate a virtual environment:
3. python3 -m venv /opt/airflow/venv
4. source /opt/airflow/venv/bin/activate
5. ``` :contentReference[oaicite:5]{index=5}.

**3.2 Installing Airflow**

1. Pin the Airflow version in requirements.txt:
2. apache-airflow[celery,postgres,crypto]==2.11.0
3. apache-airflow-providers-celery>=3.3.0
4. Install via pip:
5. pip install --constraint "https://raw.githubusercontent.com/apache/airflow/constraints-2.11.0/constraints-3.8.txt" -r requirements.txt
6. ``` :contentReference[oaicite:6]{index=6}.

**3.3 Initialize the Database**

airflow db upgrade

airflow users create --username admin --firstname Admin --lastname User --role Admin --email admin@example.com

``` :contentReference[oaicite:7]{index=7}.

## 4. Configuration for Production

### 4.1 Core `airflow.cfg` Settings

- \*\*Executor\*\*:

```ini

[core]

executor = CeleryExecutor

parallelism = 64

dag\_concurrency = 16

max\_active\_runs\_per\_dag = 16

``` :contentReference[oaicite:8]{index=8}.

- \*\*Scheduler\*\*:

```ini

[scheduler]

job\_heartbeat\_sec = 5

scheduler\_heartbeat\_sec = 5

max\_threads = 2

``` :contentReference[oaicite:9]{index=9}.

- \*\*Secrets Backend\*\*: integrate HashiCorp Vault or AWS Secrets Manager for credentials:

```ini

[secrets]

backend = airflow.providers.hashicorp.secrets.vault.VaultBackend

``` :contentReference[oaicite:10]{index=10}.

### 4.2 Logging

Configure remote logging to S3/GCS or Elasticsearch:

```ini

[logging]

remote\_logging = True

remote\_log\_conn\_id = my\_s3

remote\_base\_log\_folder = s3://my-airflow-logs

``` :contentReference[oaicite:11]{index=11}.

## 5. High Availability and Executors

### 5.1 CeleryExecutor Cluster

- \*\*Workers\*\*: run `airflow celery worker` on multiple nodes.

- \*\*Flower\*\*: monitor Celery with `pip install flower` and `airflow celery flower`.

- Ensure all workers share the same DAGs folder and `airflow.cfg` settings :contentReference[oaicite:12]{index=12}.

### 5.2 Scheduler HA

- Run two schedulers; only one actively schedules (others in standby).

- Use leader-election via database row lock or keepalive files.

## 6. Security Hardening

1. \*\*Enable RBAC\*\* in `airflow.cfg`:

```ini

[webserver]

rbac = True

authenticate = True

auth\_backend = airflow.contrib.auth.backends.password\_auth

``` .

2. \*\*Enable SSL/TLS\*\* for webserver and broker connections.

3. \*\*Restrict access\*\* with firewall rules and VPN.

## 7. DAG Synchronization and CI/CD

- Store DAGs in Git; deploy via:

- Shared NFS/SMB mount

- Periodic pull using cron/Ansible

- CI/CD pipelines pushing via `airflow dags backfill` :contentReference[oaicite:14]{index=14}.

## 8. Monitoring, Logging, and Alerting

- \*\*Metrics\*\*: expose Prometheus metrics via `apache-airflow-exporter`.

- \*\*Dashboards\*\*: use Grafana to visualize scheduler latency, task durations.

- \*\*Logs\*\*: aggregate with ELK/EFK stacks and configure alerts on failures.

## 9. Scaling, Backups, and DR

- \*\*Scaling\*\*: add more Celery workers or switch to `KubernetesExecutor`.

- \*\*Backups\*\*: schedule `pg\_dump` of metadata DB and snapshot of logs/DAGs.

- \*\*Disaster Recovery\*\*: rehearse restoring the DB and redeploying Airflow components.

---

This step-by-step guide should equip you to deploy Airflow on-premise in a production-grade, scalable, and secure manner.

::contentReference[oaicite:15]{index=15}