# ITS WORKING analyze all the steps from thebegining whch are working for me make a final result clearly including commands also

Here is a clear, final step-by-step summary of how you successfully set up pgAgent on your Red Hat Enterprise Linux 10 AWS EC2 server with PostgreSQL 17 to schedule regular backups using `pg_basebackup`. This includes all working steps and relevant commands:

### Final Setup Summary: Scheduled PostgreSQL Backups with pgAgent on RHEL 10 EC2

#### 1. Install pgAgent Package

You confirmed pgAgent 17 package installed on your server:

```
sudo dnf install pgagent_17
```

Check installation:

```
sudo dnf list installed | grep pgagent
```

#### 2. Locate pgAgent Executable

The package installed the executable as `/usr/bin/pgagent_17` (not simply `pgagent`):

```
rpm -ql pgagent_17 | grep pgagent
```

Shows:

```
/usr/bin/pgagent_17
/usr/lib/systemd/system/pgagent_17.service
```

### 3. Create Systemd Service File for pgAgent

Create `/etc/systemd/system/pgagent.service` with these contents, using the correct executable path:

```
[Unit]
Description=pgAgent PostgreSQL Job Scheduler
After=network.target postgresql.service

[Service]
User=postgres
ExecStart=/usr/bin/pgagent_17 host=localhost dbname=postgres user=pgagent
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

- Replace `dbname=postgres` if your pgAgent jobs run on a different database.

Reload systemd and start pgAgent:

```
sudo systemctl daemon-reload
sudo systemctl enable pgagent
sudo systemctl start pgagent
sudo systemctl status pgagent
```

### 4. Set Up PostgreSQL User and pgAgent Extension

Connect to PostgreSQL as superuser and run:

```
CREATE USER pgagent WITH PASSWORD 'your_secure_password';
GRANT ALL PRIVILEGES ON DATABASE postgres TO pgagent;
\c postgres
CREATE EXTENSION pgagent;
```

### 5. Configure Passwordless Access for pgAgent

Create `.pgpass` file for `postgres` user (or the user pgAgent runs as) with the replication user password:

```
sudo -u postgres bash -c "echo '127.0.0.1:5432:*:postgres:your_postgres_password' > ~/.pg
sudo -u postgres chmod 600 ~/.pgpass
```

This ensures pgAgent can connect non-interactively.

### 6. Verify Backup Directory Permissions

Ensure the backup directory exists and is writable by `postgres` user:

```
sudo mkdir -p /var/lib/pgsql/agent_backups
sudo chown -R postgres:postgres /var/lib/pgsql/agent_backups
sudo chmod 700 /var/lib/pgsql/agent_backups
```

### 7. Run Backup Manually to Verify

Manually tested this command which works and creates backups under `/var/lib/pgsql/agent_backups`:

```
pg_basebackup -h 127.0.0.1 -p 5432 -U postgres -D /var/lib/pgsql/agent_backups -Fp -P -v
```

Sample output confirmed backup completed successfully.

### 8. Create pgAgent Job in pgAdmin to Schedule Backup

In pgAdmin:

- Connect to your database.
- Under pgAgent Jobs, create a new Job (`daily_backup`).
- Add a Job Step named `backup_step` with:

```
/usr/bin/pg_basebackup -h 127.0.0.1 -p 5432 -U postgres -D /var/lib/pgsql/agent_backups -
```

- Add a Schedule, e.g., daily at 2 AM or as desired.
- Save and enable the job.

### 9. Troubleshooting Tips

- Ensure absolute paths used in pgAgent jobs.
- Verify `.pgpass` configured for non-interactive authentication.
- Check pgAgent service logs:

```
sudo journalctl -u pgagent -f
```

- Check PostgreSQL logs for replication or connection issues.

This complete setup reliably runs regular automated backups with pgAgent on your Red Hat Enterprise Linux 10 EC2 server running PostgreSQL 17.

If you want, I can help you generate any scripts or configuration files to automate this setup further.